Robust Distributed Model Predictive Control Strategies of Chemical Processes

by

Walid Al-Gherwi

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Doctor of Philosophy
in
Chemical Engineering

Waterloo, Ontario, Canada, 2010

I hereby declare that I am the sole author of this thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I understand that my thesis may be made electronically available to the public.

ABSTRACT

This work focuses on the robustness issues related to distributed model predictive control (DMPC) strategies in the presence of model uncertainty. The robustness of DMPC with respect to model uncertainty has been identified by researchers as a key factor in the successful application of DMPC.

A first task towards the formulation of robust DMPC strategy was to propose a new systematic methodology for the selection of a control structure in the context of DMPC. The methodology is based on the trade-off between performance and simplicity of structure (e.g., a centralized versus decentralized structure) and is formulated as a multi-objective mixed-integer nonlinear program (MINLP). The multi-objective function is composed of the contribution of two indices: 1) closed-loop performance index computed as an upper bound on the variability of the closed-loop system due to the effect on the output error of either set-point or disturbance input, and 2) a connectivity index used as a measure of the simplicity of the control structure. The parametric uncertainty in the models of the process is also considered in the methodology and it is described by a polytopic representation whereby the actual process's states are assumed to evolve within a polytope whose vertices are defined by linear models that can be obtained from either linearizing a nonlinear model or from their identification in the neighborhood of different operating conditions. The system's closed-loop performance and stability are formulated as Linear Matrix Inequalities (LMI) problems so that efficient interior-point methods can be exploited. To solve the MINLP a multi-start approach is adopted in which many starting points are generated in an attempt to obtain global optima. The efficiency of the

proposed methodology is shown through its application to benchmark simulation examples. The simulation results are consistent with the conclusions obtained from the analysis. The proposed methodology can be applied at the design stage to select the best control configuration in the presence of model errors.

A second goal accomplished in this research was the development of a novel online algorithm for robust DMPC that explicitly accounts for parametric uncertainty in the model. This algorithm requires the decomposition of the entire system's model into N subsystems and the solution of N convex corresponding optimization problems in parallel. The objective of this parallel optimizations is to minimize an upper bound on a robust performance objective by using a time-varying state-feedback controller for each subsystem. Model uncertainty is explicitly considered through the use of polytopic description of the model. The algorithm employs an LMI approach, in which the solutions are convex and obtained in polynomial time. An observer is designed and embedded within each controller to perform state estimations and the stability of the observer integrated with the controller is tested online via LMI conditions. An iterative design method is also proposed for computing the observer gain. This algorithm has many practical advantages, the first of which is the fact that it can be implemented in real-time control applications and thus has the benefit of enabling the use of a decentralized structure while maintaining overall stability and improving the performance of the system. It has been shown that the proposed algorithm can achieve the theoretical performance of centralized control. Furthermore, the proposed algorithm can be formulated using a variety of objectives, such as Nash equilibrium, involving interacting processing units with local objective functions or fully decentralized control in the case

of communication failure. Such cases are commonly encountered in the process industry. Simulations examples are considered to illustrate the application of the proposed method.

Finally, a third goal was the formulation of a new algorithm to improve the online computational efficiency of DMPC algorithms. The closed-loop dual-mode paradigm was employed in order to perform most of the heavy computations offline using convex optimization to enlarge invariant sets thus rendering the iterative online solution more efficient. The solution requires the satisfaction of only relatively simple constraints and the solution of problems each involving a small number of decision variables. The algorithm requires solving N convex LMI problems in parallel when cooperative scheme is implemented. The option of using Nash scheme formulation is also available for this algorithm. A relaxation method was incorporated with the algorithm to satisfy initial feasibility by introducing slack variables that converge to zero quickly after a small number of early iterations. Simulation case studies have illustrated the applicability of this approach and have demonstrated that significant improvement can be achieved with respect to computation times.

Extensions of the current work in the future should address issues of communication loss, delays and actuator failure and their impact on the robustness of DMPC algorithms. In addition, integration of the proposed DMPC algorithms with other layers in automation hierarchy can be an interesting topic for future work.

ACKNOWLEDGEMENTS

First of all I would like to express my deep appreciation and thanks to my academic supervisors Professor Hector Budman and Professor Ali Elkamel for their valuable guidance, support, patience, and kindness. Thank you Hector for your amazing supervision during the course of this work and of being always available whenever I needed your help even during weekends and when you were having your breakfast. Thank you Ali for your support and help especially in explaining optimization concepts. For both of you thank you for your wonderful friendship and it was a great pleasure to work with you.

I would like to thank the members of my PhD examining committee: Professor Fraser Forbes, Professor Fakhreddine Karray, Professor Eric Croiset, and Professor William Anderson for agreeing to serve in my exam and providing their valuable comments and suggestions.

I would like also to thank the Al-Fateh University, Tripoli, Libya and the Libyan Secretariat of High Education for the financial support. Partial financial support from University of Waterloo is also appreciated.

My thanks also go to my colleagues and professors in the Chemical Engineering program at the University of Waterloo for the friendly environment that I have really enjoyed. My thanks also go to my friends at the Process Systems Engineering Group: Rosendo Diaz, Jazdeep Mandur, Luis Ricardez, Mukesh Meshram, Ian Washington,

Mohamed Alsaleh, Khaled Elqahtani, Yousef Saif, and Penny Dorka. I do not forget to thank my best friends Mohamed Bin Shams and Ali Omer.

Last but not the least; I would like to express deep appreciation to my beloved wife, Hana, for her support, understanding, patience, and encouragement and to my joy of life Hani and Wala.

DEDICATION

To my parents: to my father who passed away before seeing his dream come true and to my mother who sacrifices without limits. To both of them I will be indebted to the rest of my life.

TABLE OF CONTENTS

LIST OF TABLES

List of Figures

ACRONYMS

CSTR	Continuous Stirred Tank Reactor
DMPC	Distributed Model Predictive Control
LMI	Linear Matrix Inequalities
MINLP	Mixed Integer Nonlinear Programming
MPC	Model Predictive Control
PID	Proportional-Integral-derivative
RDMPC	Robust Distributed Model Predictive Control

CHAPTER 1

INTRODUCTION

1.1 Background

Model predictive control MPC is a widely accepted technology for the control of multivariable processes in the process industry (Camacho and Bordons, 2003; Qin and Badgwell, 2003). The term MPC generally includes a class of algorithms that employs a dynamic model to predict the future behavior of the process, and explicitly handles process constraints and variable interactions. At each control interval, a cost function is minimized based on future response predictions, in order to obtain an optimal control trajectory. The control input corresponding to the first control interval is implemented, and the calculation procedure is repeated in the next interval to account for feedback from the process measurements. In addition, MPC can account for time delays, constraints and process interactions.

Since the advent of MPC, process industry has witnessed a transition from conventional multi-loop PI control systems to centralized MPC. The use of one centralized MPC configuration is often considered impractical due to several factors such as large computational effort required to complete the calculations in real time when many inputs and outputs are involved, sensitivity towards model errors, and low resilience in the face of equipment failures and partial shutdowns. Therefore, most industrial applications implement a decentralized MPC structure in which each MPC of

smaller dimensions than the overall process, in terms of inputs and outputs, is applied to a unit in the plant and works independently from the other controllers by optimizing local objectives and by neglecting, within the optimization, the interactions among the units. However, when the interactions are significant, this implementation leads to deterioration in overall performance and optimality of the plant, and may also jeopardize the stability of the entire system (Skogestad 2000; Rawlings and Stewart 2008). To overcome this problem, researchers have proposed Distributed MPC (DMPC) where the benefits from using the decentralized structure are preserved while the plant-wide performance and stability is improved via coordination among the smaller-dimensional controllers. Recognizing the importance of this topic, the European Commission is currently funding a 3-year project on hierarchical and DMPC with collaboration of several major European universities (*http://www.ict-hd-mpc.eu/*).

Chemical plants are composed of a network of interconnected units. These units interact with each other due to the exchange of material and energy streams. The degree of interaction depends on the dynamic behavior of the process and the geographical layout of the plant. In order to account for these interactions and to improve the performance of distributed MPC strategies researchers the use of some form of coordination between the MPC controllers for the different subsystems have been proposed (Rawlings and Stewart 2008; Scattolini 2009). In the literature, there are two types of distributed MPC strategies that take into account the interactions between the subsystems. The first type coordinates the controllers by means of a communication network through which all the MPC agents share and exchange their prediction

trajectories and local solutions and the overall solution is based on Nash optimality concepts (Due *et al.*, 2001; Li *et al.,* 2005). The iterative solution in this type of strategy reaches a Nash equilibrium point provided that some convergence condition is satisfied. However, the solution is not necessary equal to the centralized optimal solution because MPC problems with local rather than one global objective function are solved.

The second type of DMPC strategy is referred to as either a feasible cooperative strategy (Venkat, 2006) or networked MPC with neighborhood optimization (Zhang and Li, 2007). For this type of strategy a global objective function which consists of the convex sum of the local cost functions of all the subsystems is used. The solution can achieve the global optimal control decision similar to that obtained by centralized MPC if convergence is satisfied.

Common to the aforementioned coordination strategies is that they require exact knowledge of the process models to provide the designed optimal or near optimal closed loop performance and they do not address robustness in the presence of model uncertainties. Since in most industrial MPC applications linear models are used for predictions, these are never accurate due to nonlinearity or inaccurate identification. One alternative to mitigate this problem is to use nonlinear models for prediction but with such models it is very difficult to theoretically prove stability and performance. Thus, the robustness of distributed control strategies to model error has been identified as one of the major factors for the successful application of distributed MPC strategies (Rawlings and Stewart, 2008). In regards to the coordination problem, the frameworks proposed in

the literature rely on feedback only to account for plant-model mismatch and do not explicitly consider the robustness issues. Therefore developing new online algorithms that explicitly consider model errors is of great importance from the theoretical and practical viewpoints.

Another major challenge for the application of DMPC is the selection of the control structure to be used in the distributed strategy. This involves the selection of which manipulated variables, controlled variables and states are assigned to each subsystem for which an MPC controller is to be applied. Despite the fact that a significant number of publications have appeared in the literature dealing with distributed MPC there is no systematic methodology to select the best control structure (Scattolini 2009). The problem is generally decomposed in an ad hoc fashion and based on engineering insights. Mercangöz and Doyle (2007) extended a heuristic procedure of partitioning reported by Vadigepalli and Doyle (2003) to distributed MPC. It should also be recognized that the selection of the control structure will also be related to the presence of model errors as shown later in the thesis. Therefore, there is a need for developing systematic tools to select the best control structure in the context of DMPC to balance performance in the presence of model errors against simplicity (Scattolini 2009).

Following the above, in the current research work two problems related to distributed MPC strategies are considered: the selection of the control structure and the coordination problem in the presence of model errors. The following section summarizes the objectives of the current research.

1.2 Objectives of the Research

The following are the main objectives that were accomplished during the course of the current research:

- Development of a systematic methodology based on robust control tools to select the best control structure for distributed MPC strategy and at the same time to provide a performance assessment for different coordination strategies in the presence of model uncertainty. Both set-point tracking and disturbance rejection problems were considered.
- Investigation of robustness issues related to the current distributed MPC in the presence of uncertainties.
- Development of new online algorithms for robust DMPC that account for parametric model uncertainty.

1.3 Contributions of the Current Research

The robustness of DMPC strategies with respect to model uncertainty has been identified by researchers as a key factor in the successful application of DMPC. Despite the significant research available, only limited work related to the coordination of DMPC in the presence of model errors is reported in the literature. The contribution of the current research is to address robustness issues as per the research objectives listed in the previous subsection.

The thesis is organized as follows. Chapter 3 presents a new systematic methodology for the selection of a control structure in the context of DMPC. The methodology seeks for an optimal trade-off between performance and simplicity of structure (e.g., a centralized versus decentralized structure) and is formulated as a multi-objective mixed-integer nonlinear program. The multi-objective function is composed of the contribution of two indices: 1) a closed-loop performance index computed as an upper bound on the variability of the closed-loop system due to the effect of changes in set-point or disturbance on the outputs, and 2) a connectivity index used as a measure of the simplicity of the control structure. The parametric uncertainty in the models is explicitly considered in the methodology. The efficiency of the proposed methodology is shown through its application on several benchmark simulation examples.

In chapter 4, a novel algorithm for robust DMPC that explicitly accounts for parametric uncertainty in the model was developed. The algorithm requires the decomposition of the model of the entire system into N subsystems' models and the solution of corresponding N convex simultaneous optimization problems. The objective of these optimizations is to minimize an upper bound on a robust performance objective by using a time-varying state-feedback controller for each subsystem. Model uncertainty is explicitly considered through the use of a polytopic model. Based on this polytopic representation the algorithm employs a linear matrix inequality (LMI) approach, in which the solution is obtained in polynomial time. Therefore, the algorithm can be implemented in real-time control applications and thus has the benefit of enabling the use of a decentralized structure while maintaining overall robust stability and robust performance.

It is shown that the proposed algorithm can achieve in the limit the theoretical performance of centralized control. Furthermore, the proposed algorithm can be formulated for a variety of optimization objectives, such as Nash equilibrium objective. Nash equilibrium is of practical value since it may address the situation where different interconnected units are operated by different owners with their own optimization goals.

In chapter 5, the main goal is to improve the online computational efficiency of robust MPC. To achieve this objective, a dual-mode control approach is proposed in which the control action is composed of the contribution of state feedback and a set of additional degrees of freedom. The state feedback calculations that are more time demanding are solved off line whereas the additional degrees of freedom are solved on line through a quick LMI calculation thus rendering the online solution more efficient. A relaxation method was incorporated within this algorithm to satisfy an initial feasibility constraint by introducing slack variables that converge quickly to zero. Simulated case studies have illustrated the applicability of this approach and have demonstrated the significant improvement in computation time that can be achieved with this algorithm as compared to the algorithm proposed in chapter 4.

Different results of this research have been presented in some publications as well as several oral presentations at conferences and meetings as follows:

Refereed Publications

Walid Al-Gherwi, Hector Budman, and Ali Elkamel, "Selection of control structure for distributed model predictive control in the presence of model errors," *J. Process Control,* (20) 270-284, 2010.

Walid Al-Gherwi, Hector Budman, and Ali Elkamel, "An online algorithm for robust distributed model predictive control," *Proceedings of the Advanced Control of Chemical Processes* ADCHEM, Paper no. 33, 6 pages, 2009.

Walid Al-Gherwi, Hector Budman, and Ali Elkamel, "Robustness issues related to the application of distributed model predictive control strategies," *Proceedings of the 17th World Congress, the International Federation of Automatic Control* IFAC, 8395 – 8400, 2008.

Walid Al-Gherwi, Hector Budman, and Ali Elkamel, "A robust distributed model predictive control algorithm," (Submitted to *Ind. & Eng. Chem. Res*), 2010.

Oral Presentations at Conferences & Meetings

Walid Al-Gherwi "Recent developments in distributed model predictive control", *A seminar presented at the Faculty of Engineering University of Regina,* Regina, SK June 2010.

Walid Al-Gherwi, Hector M. Budman, Ali Elkamel, "Robust distributed model predictive control strategies", *Annual meeting of Control and Stats,* Waterloo, ON May 2010.

Walid Al-Gherwi, Hector Budman, and Ali Elkamel, "The role of convex optimization with Linear Matrix Inequalities in the application of DMPC," *CORS-INFORMS International*, Toronto, Canada, June 2009.

Walid Al-Gherwi, Hector M. Budman, Ali Elkamel, "Robustness issues related to the application of distributed model predictive control strategies", *The 17th IFAC World Congress*, Seoul, Korea, July 2008.

Walid Al-Gherwi, Hector M. Budman, Ali Elkamel, "Distributed model predictive control", *Annual meeting of Control and Stats,* Montreal, QC May 2008.

Walid Al-Gherwi, Hector M. Budman, Ali Elkamel, "Optimal selection of decentralized MPC structure for interconnected chemical processes", *57th Canadian Chemical Engineering Conference*, Edmonton, Alberta, October 2007.

Walid Al-Gherwi, Hector M. Budman, Ali Elkamel, "Robust control tools for distributed model predictive control: Model Decomposition and Coordination", *South Western Ontario Operational Research Day,* Waterloo, ON, October 2007.

CHAPTER 2

LITERATURE SURVEY

2.1 Model Predictive Control

Model predictive control (MPC) is a widely accepted technology for the control of multivariable processes in the chemical industry. There are many successful applications that have been reported in the literature (Qin and Badgwell, 2003). Nowadays, applications of MPC are also reported for other processes ranging from robots and automotive to power plants (Camacho and Bordons, 2003).

MPC refers to a family of control algorithms that utilize an explicit dynamic model of the process to predict its future behavior and solve for optimal control moves by minimizing an objective function based on an output prediction. Since the prediction is based on a model, the latter is the cornerstone of MPC and therefore the type of MPC algorithm to be used depends on the type of model chosen. Step response, impulse response, transfer function, and state-space models are various types of linear models used in MPC algorithms. MPC algorithms can also employ nonlinear models but such nonlinear predictive algorithms will not be considered in the current work since they are less common in industrial practice and are more difficult to analyze for stability and performance. The objective function chosen for MPC can be either linear or quadratic but in most MPC algorithms the latter is widely used since it provides better error averaging properties and an explicit analytical solution can be easily obtained for the special case of

control without constraints whereas quadratic programming can be used to solve the constrained case.

Figure 2.1 illustrates the general methodology of all classes of MPC. At each control interval, the output behavior of the plant is predicted over the prediction horizon using the process model. Then the set of control actions over a predefined control horizon is obtained by minimizing an objective function. The changes in control actions are assumed to be zero beyond the control horizon. Only the first value in the set of calculated control actions is implemented in the process and the entire calculation is repeated again at the next control interval to account for process feedback.

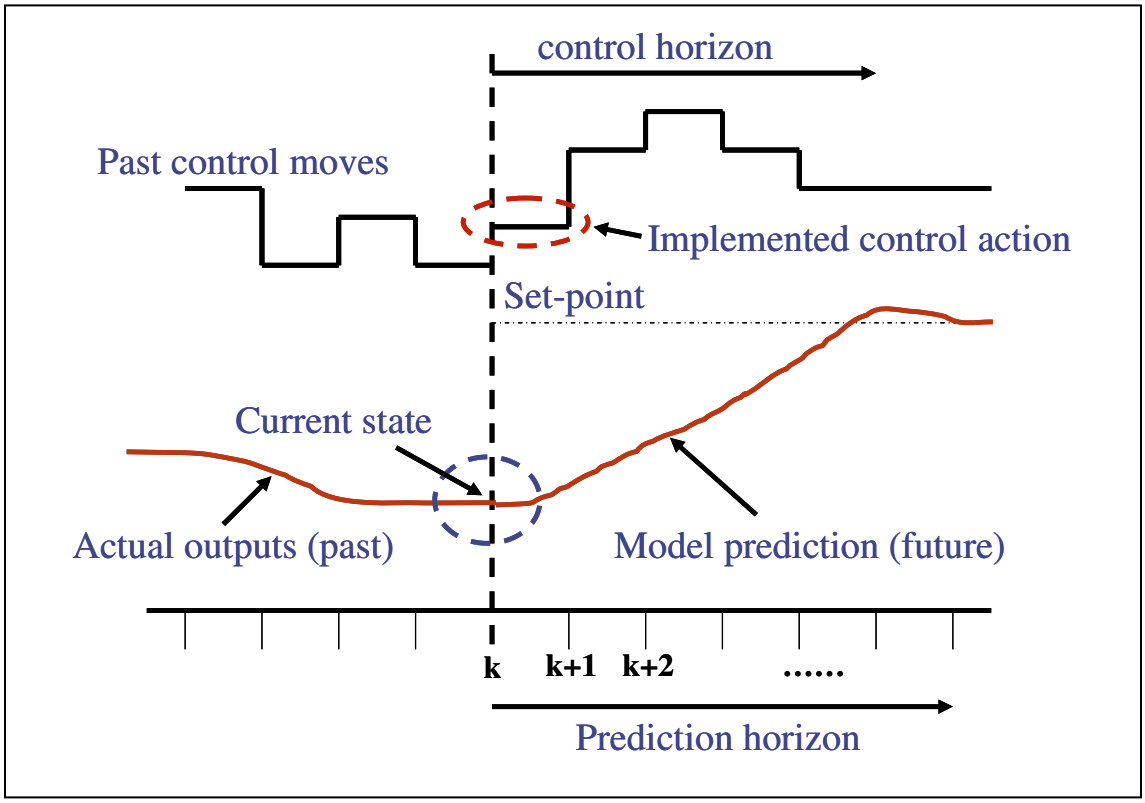


Figure 2.1 MPC methodology

Although the advent of MPC technology is originated by the pioneering work of Richalet *et al.* (1978) and Cutler and Ramaker (1979), some consider the early work of Kalman (1960) to be the precursor of MPC regarding the concept of output prediction.

Richalet *et al.* (1978) employed a linear impulse response model to represent the process whereas Cutler and Ramaker (1979) used linear step response models. In both formulations, an unconstrained quadratic objective function was considered. The optimal inputs were obtained by a heuristic iterative algorithm in the former formulation whereas in the latter formulation the inputs were obtained from the solution of a least-squares problem. The MPC formulation proposed by Cutler and Ramaker (1979), which is known in the literature as Dynamic Matrix Control (DMC), was extended to handle constrains on process variables. The constrained algorithm is usually referred to as Quadratic Dynamic Matrix Control (QDMC) to indicate the use of a quadratic objective function (Cutler *et al.*, 1983); (Garcia and Morshedi, 1986). If the constraints are linear then the resulting optimization problem solved for QDMC is convex.

Generally, the step response and impulse response models require an excessive number of coefficients (typically between 30 to 50) in order for the MPC to achieve good performance (Lunstrom *et al.*, 1995) and this large number of coefficients is typically related to the settling time of the process to be controlled. This is considered as a limitation for both DMC and QDMC algorithms since many multivariable processes require a large number of coefficients resulting in intensive calculations needed for the optimization. To circumvent this limitation, researchers have proposed the use of state-

space models that can potentially save memory compared to the input-output models mentioned above. In addition, a rich theory is available for linear state-space models that can be used to simplify the numerical solutions, and for testing the controllability, observability, and stability of the system (Aplevich, 2000). Li *et al.* (1989) presented a state-space based MPC form based on step response models to implement a DMC algorithm. Prett and Garcia (1988) replaced the step response model with a general discrete state-space model and therefore the effect of truncation errors caused by using step coefficients was removed. Muske and Rawlings (1993) developed a linear MPC based on sate-space models to control stable and unstable systems. They showed that the proposed algorithm can be made equivalent to an infinite horizon regulator by incorporating a terminal cost term within the cost function to be optimized. A comprehensive MPC formulation based on discrete state-space models is reported by Maciejowski (2002).

2.3 Distributed Model Predictive Control (DMPC)

> *Centralization often means accessing an entire operation (or production line) by one person or a small group of people from a single point. Such complete surrender to one or a few pieces of hardware can be considered putting all your eggs in one basket. If that is your choice, you'd better watch that basket! (Mark Twain summarized that wisdom).*
>
> (Scheiber, 2004)

Since the advent of MPC technology, process industry has witnessed a shift from conventional multi-loop decentralized PI control strategies to centralized multivariable MPC strategies. The ability of MPC to handle process constraints and its direct application to multivariable systems attracted practitioners to implement MPC technology. However, centralized multivariable control is often considered impractical due to drawbacks such as the high computational effort required when dealing with processes with relatively large number of inputs and outputs, the need to obtain an expensive multivariable dynamic model to represent the entire process or plant to be controlled, sensitivity to model errors and to changes in operating conditions, and its low resilience with respect to partial equipment failure or partial plant shutdown (Skogestad, 2004; Venkat, 2006). This led to the idea of partitioning the original process into smaller units or subsystems and the application of MPC controllers to each one of these subsystems. The operations of several MPC controllers in such fashion have been referred to as decentralized MPC. On the other hand, although decentralized MPC applications could result in less computations, when the individual MPC controllers for the different subsystems are operated in a completely decentralized fashion closed loop performance may be significantly hampered since some or all of the interactions are ignored and the controllers may become unstable if these interactions are strong. As a remedy to this problem, researchers have proposed the use of some form of coordination between the MPC controllers for the different subsystems by allowing the controllers to exchange information via a devoted communication network. Coordination strategies based on Nash equilibrium (Li *et al.,* 2005) or cooperative schemes based on weighted cost functions (Venkat, 2006) have been reported. These strategies have different

information structure which describes the way of transferring information among the subsystems and their assigned controllers (Lunze, 1992). Figure 3.1 illustrates a general structure and the mode of information transfer in different MPC strategies; *viz.*, centralized, fully decentralized, and coordination-based. Centralized MPC requires a centralized dynamic model to represent the entire process and the availability of complete sensor information. The optimal control moves are obtained by minimizing a cost function (objective function) that includes all the controlled variables. Since the centralized control takes into account the interactions within the system, it is theoretically expected to result in the best achievable performance provided that the model is perfect. In contrast, for distributed MPC (DMPC), either fully decentralized or coordinated, each control agent only uses a local dynamic model and has access to local measurements. In fully decentralized MPC, the interactions between the subsystems are totally ignored and each MPC has access to local measurements and solves a local cost function that includes only the controlled variables assigned to the specific subsystem without considering the solutions of the other controllers. On the other hand, in coordinated MPC the controllers have knowledge about the interactions through the use of interactive models (local + interaction terms) and by sharing information and combining the solutions to the local minimization problems to achieve a global objective. DMPC therefore has the flexibility related to its decentralized structure while keeping the ability to achieve performance that can be close to that of centralized control by accounting for process interactions. Further details on computational issues related to the three strategies are discussed later in this chapter.

In light of the previous introduction, two questions are posed; Firstly, what is the best control structure that will provide an optimal trade-off between closed-loop performance and simplicity? Secondly, how is it possible to address model uncertainty and the robustness of DMPC strategies in the presence of plant-model mismatch?

During the development of this research work two comprehensive survey papers related to DMPC have been already published where the two questions posed above were put forward as key open issues in DMPC research. Rawlings and Stewart (2008) in their discussion of challenges in DMPC technology emphasized the importance of addressing robustness issues and the need to develop robust DMPC strategies in the presence of model errors. Scattolini (2009) has recently provided a comprehensive review on DMPC strategies and urged for the development of analysis tools to reach an optimal trade-off between performance and structure simplicity. The objective of the following sections is to provide a review of DMPC strategies previously proposed as a preamble to the research conducted in this thesis.

2.3.1 Distributed MPC Structure

The main goal of system decomposition is to partition the original problem into smaller subsystems of manageable size (Lunze, 1992) and to find the control structure that interconnects these subsystems (Skogestad, 2000). Skogestad (2000) reported that the controller structure decompositions can be classified as either; decentralized (or horizontal) decomposition, or hierarchical decomposition. The decentralized

decomposition is mainly based on the process units and therefore on the physical structure whereas the hierarchical decomposition is based on process structure, control objectives, and time scale. The decentralized decomposition consists of breaking the original system down into smaller subsystems that are independent of each other due to either weak interaction (coupling) among them or simply because the interactions are ignored for control design (Skogestad, 2000; Negenborn *et al.*, 2004). However, chemical processes are often composed of networks of interconnected units that interact with each other due to exchange of material and energy streams and completely neglecting these interactions may often lead to loss in control performance. On the other hand, the hierarchical decomposition considers that the subsystems depend on each other and takes into account the interactions (Skogestad, 2000; Negenborn *et al.*, 2004; Venkat, 2006). Although there is a significant research that has focused on DMPC in the recent years, a very clear gap in the literature regarding the decomposition problem can be noticed. Negenborn *et al.* (2004) indicated in their survey that there is no reported generic method to obtain such structure. In general, the available methods for control structure in the context of DMPC assume that a centralized model of the system is available and that this model can then be partitioned or decomposed into several subsystems using either engineering insights or structural properties of the mathematical model (Vadigepalli and Doyle, 2003; Mercangöz and Doyle, 2007).

Motee and Sayyar-Rodsari (2003) proposed an algorithm for optimal partitioning in distributed MPC. An open loop performance metric is weighted against the closed loop cost of the control action for the system in order to obtain optimal grouping of the

system. An unconstrained distributed MPC framework was used and then a weighting matrix was defined to convert the distributed system to a directed graph. However, the effect of model errors (plant-model mismatch) on the decomposition was not explicitly considered in the algorithm. Furthermore, they did not consider the problem of simplifying the communication structure which is one of the sought objectives when applying DMPC.

Vadigepalli and Doyle (2003) reported a semi-automatic approach to decompose the overall system model into interacting subsystems for distributed estimation and control. A heuristic procedure was provided to guide the decomposition based on analysis of the mathematical model and the information about the plant topology (flowsheet). The basic idea behind this decomposition method is that some slow variables can be expressed as a function of some faster variables and in that way the faster variables can be eliminated from certain state equations. The procedure is summarized as follows: the first step is to use the plant flowsheet to identify the process units and therefore to consider each unit as a subsystem after this the plant model is discretized based on a chosen sampling time. The next step is to identify the overlapping in the states resulting from the discretization step and this overlapping indicates how the subsystems are connected and provides information about the communication required. These steps are repeated in a trial-and-error manner in order to minimize the communication and accordingly the computational effort by changing the sampling time and by successively repeating the procedure. Further partitioning and/or combining of subsystems can be required. However, the effect of the resulting decomposition on the closed-loop

performance is not considered explicitly. Two chemical engineering examples were considered to illustrate the method. The same approach was extended to DMPC in the work of Mercangöz and Doyle (2007). The procedure did not consider uncertainties in the model parameters.

A plant decomposition iterative algorithm was proposed by Zhu and Henson (2002) based on the earlier work of Zhu *et al.* (2000). The basic idea is to partition the plant into linear and nonlinear subsystems according to the nonlinear properties of the corresponding subsystems and applied MPC to each subsystem. They used heuristics and a priori process knowledge to determine the relative nonlinearity of a subsystem. A styrene plant was used as a case study. The approach is out of the scope of this work since the emphasis is on the process nonlinearity and accordingly it uses nonlinear MPC technology whereas the current work considers linear MPC only.

Considering model-based control techniques, Samyudia and co-workers (1994; 1995) presented a systematic methodology for the control and design of multi-unit processing plants. The main focus of the work was to establish an approach for selecting the best decomposition for decentralized control design. The model of the whole plant, usually represented by a linear state-space model, is decomposed based on either physical unit operations that are interconnected together or across the units by considering the dynamics of the controlled variables even if these variables belong to different unit operations. This results in many alternative decomposition candidates for the same input-output pairings. The method utilizes the gap metric and normalized coprime factorization

concepts of robust control theory. These indicators are used to determine the best system decomposition strategy so that the overall stability and achievable performance can be examined by observing the indicators. It is concluded that individual controller complexity is less important than the plant decomposition strategy and that decomposition based on the physical unit operations does not always produce better performance than model-based decomposition. The methodology searches for one best plant decomposition at a specific operating point. As an extension to this work, Lee *et al.* (2000) obtained the best decomposition subregions in an operating space and these subregions are represented by a grid of linear models obtained from linearizations around the operating conditions that correspond to each point on the grid. The decomposition and controller design are carried out in two different steps and consequently, open-loop information is used in the selection of best decomposition. The related research work presented several case studies from the chemical process industry. The types of model decomposition suggested are considered in the next chapter.

Relevant results related to the computational aspects and coordination schemes in DMPC from the literature are reviewed. As it has been mentioned earlier, surveys that define the new opportunities and challenges associated with coordinating DMPC agents can be found in (Negenborn *et al.*, 2004; Rawlings and Stewart, 2008; Scattolini 2009). The information structure of the system is basically the most important element that determines which type of coordination should be used for a particular application (Camponogara *et al.* 2002). Referring back to Figure 3.1, classifying DMPC strategies with coordination into two categories referred to as; communication-based (Nash-based)

and feasible-cooperative control (Venkat, 2006; Rawlings and Stewart, 2008), there are a total of four possible types of MPC schemes that can be considered. Table 2.1 summarizes the main computational requirements and model structure for each type.

Table 2.1 MPC Coordination Choices

Type	Model	Objective Function	Result
Centralized	centralized model of the entire process	One overall objective for the system	Optimal nominal performance is achieved
Fully Decentralized	Independent local model for each subsystem (interactions are ignored)	Independent local objective for each subsystem	Loss in optimal performance is expected and could be significant
Communication or Nash - based	Interaction models are considered along with local models.	Local objective for each subsystem. Cooperative scheme via communication.	The entire system will arrive at Nash equilibrium if convergent condition is satisfied
Feasible-cooperative	Interaction models are considered along with local models.	Local objective for each subsystem is composed of the sum of all other objectives.	Can achieve the centralized performance when the convergence is reached.

Most of distributed MPC approaches available in the literature adopt the communication-based coordination which results in Nash optimality. For numerical convenience or if there are constraints the solution is achieved iteratively where at each

iteration step the interaction information is shared among the subsystems and their local objectives are solved until convergence is achieved provided that a feasible solution exists. The equilibrium poit thus achieved is referred to as a Nash equilibrium which is, in the case of DMPC, the intersection of the control actions of all MPC controllers in the system (Negenborn *et al.*, 2004; Venkat, 2006; Rawlings and Stewart, 2007). However, a loss in performance is expected since Nash-based solution is not necessary equal to the centralized solution since the corresponding objective functions of these two strategies are different. Venkat et al. (2006) argued that, when wrong (-bad) input-output pairings are selected, communication only cannot guarantee either optimality or the stability of the system and following these arguments he developed the feasible-cooperative approach that achieves the centralized MPC solution when convergence is reached. Bad pairings are referred to those ones that, if selected for control, will exhibit poor closed loop performance based on RGA (Relative Gain Array) considerations (Bristol, 1966). Additional details regarding RGA and input-output pairings are given in section (2.4) of this chapter. Zhang and Li (2007) showed that for unconstrained distributed MPC the performance is equal to the centralized solution. The following paragraphs are a review of literature coordination methods presented in chronological order.

Xu and coworkers (1988) discussed an algorithm for decentralized predictive control based on Nash-based approach. A step-response model is used for modeling. An analysis of the stability and performance of the system is presented. Robustness of the algorithm in face of model errors was not addressed.

Charos and Arkun (1993) showed how the QDMC problem can be decomposed into smaller and less computationally demanding sub-problems which can be solved in a decentralized manner. Simulation examples and CPU time requirements were presented for comparison purposes.

Katebi and Johnson (1997) proposed a decomposition-coordination scheme for generalized predictive control. A high level coordinator was used to iteratively find an optimal solution. Perfect models were assumed in their study thus robustness to model error was not considered.

Applying neural networks based predictive control, Wang and Soh (2000) proposed an adaptive neural model-based decentralized predictive control for general multivariable non-linear processes. The proposed method was applied to a distillation column control problem. They noticed that a loss in performance can occur when the interactions are strong. Large training data sets were also required for the proposed technique to get acceptable results.

Jia and Krogh (2001) explored a distributed MPC strategy in which the controllers exchange their predictions by communication to incorporate this information in their local policies. In another work, Jia and Krogh (2002) proposed a min-max distributed MPC method that treats the interactions as bounded uncertainties.

Based on Nash optimality, Du *et al.* (2001) presented an algorithm for DMPC based on step-response models. A closed form solution was developed for the unconstrained case and the existence of the solution was analyzed. The solution formulation reported in that work is extended to state-space models in the current research work and it is explained in details in the next chapter.

Camponogara *et al.* (2002) discussed the distributed MPC problem and reported an algorithm for cooperative iteration. In addition, heuristics for handling asynchronous communication problems were provided and the stability of distributed MPC was studied. A power system application was presented as a case study.

In an application to multi-vehicle system, Dunbar (2005) reported distributed-cooperative formulation for dynamically coupled nonlinear systems. One drawback in this theoretical formulation is the requirement that at least ten agents have to be considered to guarantee stability.

In a continuation of the previous work of Du *et al.* (2001), Li *et al.* (2005) applied the Nash-based algorithm to the shell benchmark problem. Also, they extended the analysis of stability and the condition for convergence to the Nash equilibrium. In addition, the stability and performance for a single-step horizon under conditions of communication failure are examined. Similar to their previous work, robustness issues related to their algorithm have not been addressed.

Venkat *et al.* (2006) developed a new distributed MPC strategy that differs from previously reported Nash equilibrium-based methods. He showed that modeling the interactions between subsystems and communicating the local predictions does not guarantee closed-loop stability. The feasible-cooperative strategy proposed in their study modifies the local objective functions by using a weighted sum of all objectives. If the iterative algorithm reached convergence the solution becomes equal to the centralized case. However, this might require several iterations and therefore an intermediate termination of the algorithm may be necessary to save computations. Several chemical engineering examples were examined to illustrate the advantages of the methodology. However, Venkat has not addressed robustness issues.

Magni and Scattolini (2006) proposed a fully decentralized MPC methodology for nonlinear systems. No exchange of information between local controllers is assumed in this study. In order to ensure stability, a conservative contraction mapping constraint is used in the formulation which might be difficult to satisfy in practice leading to very conservative controllers.

Mercangöz and Doyle (2007) proposed a distributed model predictive estimation and control framework. The heuristic approach reported in Vadigepalli and Doyle (2003) was extended for DMPC strategy. They reported that the communication among agents during estimation and control improves the performance over the fully decentralized MPC strategy and approaches the performance of centralized strategy at the nominal operating conditions. On the other hand, only one iteration was performed during each

sampling period since performing all the iterations until convergence was found to increase computational cost dramatically. An experimental four-tank system was investigated and the results showed that the computation effort was lower as compared to the computational effort required for the centralized strategy.

Two networked MPC schemes based on neighborhood optimization for serially connected systems are presented in Zhang and Li (2007). The scheme is very similar to the methodology presented in Venkat (2006) and they showed that the solution of the unconstrained version is equal to that of the centralized strategy. The analysis of convergence and stability was also presented.

Based on the Dantzig-Wolfe decomposition and a price-driven approach, a DMPC framework for steady-state target calculations was proposed by Cheng *et al.* (2007 & 2008). Recently, the approach has been extended for directly coordinating DMPC agents. Step response models were used to avoid designing state estimators and bias terms were used to account for model uncertainty (Marcos et al. 2009).

Sun and El-Farra (2008) proposed a methodology to integrate control and communication and developed a quasi-decentralized control framework in which an observer model of the entire system was used in each subsystem to provide predictions in case of any communication delay or failure. The assumption in their framework is that interactions are through the state variables and that the inputs are decoupled and therefore no iterations are required.

A coordination strategy based on a networked decentralized MPC was proposed by Vaccarini and coworkers (2009). Performance was improved by including the solutions from previous control interval which will decrease computations as well. Conditions for stability were also provided for the unconstrained case.

Xu and Bao (2009) addressed the plantwide control problem from a network perspective. The model used integrates the physical mass and energy links with information links resulting in a two-port linear time-invariant system. They applied the dissipativity theory to address stability and performance. A reactor distillation system was considered as a case study.

It should be emphasized again that all the previous formulations do not address the robustness with respect to model errors explicitly and rely on feedback to account for any mismatch. Handling uncertainty in the controller model has been identified as one of the major factors for the successful application of DMPC strategies (Rawlings and Stewart, 2008). Also, the optimal selection of the system into smaller subsystems, i.e. the decomposition problem, in face of model errors has not been addressed. In summary, the state-of-the-art DMPC strategies lack algorithms that explicitly consider model errors.

2.4 Interaction Measures

It has been mentioned previously how the decentralized control structure is desirable due to its practical advantages compared to its centralized counterpart. If the process to be controlled is a 2 × 2 system and represented by the transfer matrix $\boldsymbol{G}(s) = g_{ij}(s)$; $(i,j = 1,2)$ then the fully decentralized control system requires identifying the dominant transfer functions in $\boldsymbol{G}(s)$ and therefore ignoring either its diagonal or off-diagonal elements. In this case, two alternatives can be used as an approximation to the full system according to the following expressions:

$$\tilde{\boldsymbol{G}}_1(s) = \begin{bmatrix} g_{11}(s) & 0 \\ 0 & g_{22}(s) \end{bmatrix} \text{ or } \tilde{\boldsymbol{G}}_2(s) = \begin{bmatrix} 0 & g_{12}(s) \\ g_{21}(s) & 0 \end{bmatrix} \tag{2.1}$$

where $\tilde{\boldsymbol{G}}(s)$ is an approximation of $\boldsymbol{G}(s)$.

The key for a successful decentralized control strategy is to choose the best approximation or in other words the best pairings between manipulated and controlled variables that yield little or no loss in performance. The number of alternatives increases with the size of the process to be controlled and therefore a metric or measure is required to systematically compare the alternatives. The idea is to measure the interactions with these metrics in order to ignore the weak channels. A key goal of an interaction measure is to provide a selection criterion for the best pairings (Grosdidier and Morari, 1986; Skogestad and Postlethwaite 2005). The most widely used interaction measures or indices are briefly presented in the following paragraphs.

A simple but rather efficient measure is the relative gain array RGA developed by Bristol (1966) for the analysis of multivariable systems. This measure requires only steady-state information to measure the process interactions and provide a guideline of choosing the best input-output pairings. For square plants of size n, the relative gain array $\boldsymbol{\Lambda}$ is given by:

$$\boldsymbol{\Lambda} = \begin{array}{c} \\ y_1 \\ y_2 \\ \cdots \\ y_n \end{array} \begin{array}{c} \begin{array}{cccc} u_1 & u_2 & \cdots & u_n \end{array} \\ \begin{bmatrix} \lambda_{11} & \lambda_{12} & \cdots & \lambda_{1n} \\ \lambda_{21} & \lambda_{22} & \cdots & \lambda_{2n} \\ \cdots & \cdots & \cdots & \cdots \\ \lambda_{n1} & \lambda_{n2} & \cdots & \lambda_{nn} \end{bmatrix} \end{array} \tag{2.2}$$

where y_i and u_i are the controlled variables and manipulated variables; respectively. The entries λ_{ij} are the dimensionless relative gains between y_i and u_j and they are defined by:

$$\lambda_{ij} \triangleq \frac{\left(\partial y_i / \partial u_j\right)_u}{\left(\partial y_i / \partial u_j\right)_y} = \frac{\text{open-loop gain}}{\text{closed-loop gain}} \tag{2.3}$$

The name "relative gain" is due to the ratio between the gains defined in the above expression. This quantity is defined as a useful measure of interactions (Skogestad and Postlethwaite, 2005). The following are some of its main properties (Seborg *et al.*, 2004):

1. The sum of the elements in each row or column is equal to one which makes it normalized.

2. Scaling and choice of units do not affect the relative gains since they are dimensionless.
3. The RGA is a measure of sensitivity in the gain matrix towards element uncertainty.

The best pairings are selected based on RGA elements and as a recommendation good pairings correspond to RGA values close to one that indicate low interaction effects. On the other hand, negative RGA values indicate large interactions and possible closed loop instability is expected when inputs and outputs are paired according to these negative RGA elements pairings.

A major disadvantage of the RGA approach is that it ignores process dynamics that could be crucial in the selection of best parings. This led many researchers to extend the standard approach to consider process dynamics and develop the dynamic RGA (Grosdidier and Morary, 1986 and Skogestad and Postlethwaite, 2005). However, the dynamic RGA is not as easy to use and interpret as the standard steady–state based RGA. Regarding uncertainty in the model parameters, little attention was given to their effect on RGA. However, Chen and Seborg (2002) developed analytical expressions for RGA uncertainty bounds.

Manousiouthakis *et al.* (1986) generalized the steady-state RGA concepts to the block relative gain array BRGA. It is used in the block pairings of inputs and outputs where each block may have several inputs and outputs. A methodology was proposed for

screening alternative decentralized control structures. The development was based on the assumption of perfect control. Arkun (1987) proposed a dynamic version of BRGA. Kariwala et al. (2003) studied the BRGA, presented new properties and established its relation with closed-loop stability and interactions. They showed that systems with strong interactions can have BRGA that is close to the identity matrix and this result contradicted some of the previous results of Manousiouthakis *et al.* (1986).

A new interaction measure (μ) in the context of structured singular value SSV was developed in (Grosdidier and Morari, 1986; Grosdidier and Morari, 1987). This measure is defined for multivariable systems under feedback with diagonal or block diagonal controllers. The Structured Singular Value (SSV) analysis or μ analysis considers a plant model that is subject to unstructured or structured uncertainty. It also considers that there is an interconnection between the model and the uncertainty by means of a Linear Fractional Transformation LFT as shown in Figure 2.4.

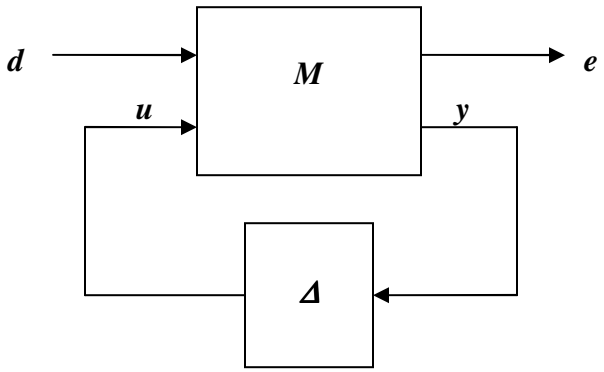


Figure 2.4 General M-Δ LFT connection

In the framework shown in figure 2.4, the linear time-invariant LTI system $\boldsymbol{M} \in \mathbf{C}^{n\times n}$ represents the controller, the nominal models of the system, sensors, and actuators. The input vector $\boldsymbol{d}$ includes all external inputs to the system such as disturbances and set-point signals whereas the vector $\boldsymbol{e}$ represents all the output signals generated by the system. $\boldsymbol{M}$ can be partitioned as follows:

$$\begin{bmatrix} \boldsymbol{e} \\ \boldsymbol{y} \end{bmatrix} = \begin{bmatrix} \boldsymbol{M}_{11} & \boldsymbol{M}_{12} \\ \boldsymbol{M}_{21} & \boldsymbol{M}_{22} \end{bmatrix} \begin{bmatrix} \boldsymbol{d} \\ \boldsymbol{u} \end{bmatrix} \tag{2.4}$$

The relationship between $\boldsymbol{e}$ and $\boldsymbol{d}$ is given by:

$$\boldsymbol{e} = \boldsymbol{F}_u(\boldsymbol{M}, \Delta)\,\boldsymbol{d} = (\boldsymbol{M}_{22} + \boldsymbol{M}_{21}\Delta(\boldsymbol{I} - \boldsymbol{M}_{11}\Delta)^{-1}\boldsymbol{M}_{12})\,\boldsymbol{d} \tag{2.5}$$

where $\boldsymbol{F}_u(\boldsymbol{M}, \Delta)$ is the upper LFT operator. Further definitions and theorems for robust stability and performance can be found in Doyle and Packard (1987).

SSV can be used to predict the stability and measure the performance loss of the decentralized control structure. In Braatz *et al.* (1996), screening tools were developed based on μ for measuring the performance in the presence of general structured model uncertainty.

In summary, RGA and BRGA are simple and useful tools for measuring interactions and screening control structure alternatives. However, they can not be easily used to test the stability and performance of the closed-loop system. On the one hand steady state RGA measures do not consider the system properties under dynamic conditions and on the other hand the dynamic RGA are made of frequency dependent vectors of gains that

are difficult to interpret and apply to practical situations. The μ-measures on the other hand result in very conservative designs when applied to state space models to be used as the basis of MPC algorithms. Furthermore, some of these measures require complex algebraic manipulation that could become more difficult when extended to the complicated structure of distributed MPC involving many manipulated and controlled variables.

2.5 Linear Matrix Inequalities (LMIs) and Robust Control

Most MPC's under operation in the chemical industry are designed based on linear models of the system. However, linear models are never accurate due to nonlinearity or inaccurate identification. Although nonlinear MPC can partially mitigate this problem, its application is more limited since it is more difficult to design for stability and performance. Therefore, nonlinear MPC is beyond the scope of the current review. Feedback control has to be designed to provide good performance in the presence of both disturbances and model errors. Robust control design refers to design methodologies that explicitly account for the plant-model mismatch in the design. Most of robust control approaches assume that there is a set or family of plants to represent the possible sources of uncertainties (Morari and Zafiriou, 1989; Camacho and Bordons, 2003). Although a significant research has been published for the design and analysis of robust MPC systems, robustness of distributed MPC strategies has not been explicitly addressed. It is worth to mention that MPC is sensitive towards model uncertainty. To illustrate such sensitivity let us consider the following multivariable control problem with

3 manipulated variables and 3 controlled variables given by the following transfer function matrix:

$$\boldsymbol{G}(s)=\begin{bmatrix} \frac{4.05e^{-6s}}{50s+1} & \frac{1.77e^{-6s}}{60s+1} & \frac{5.88e^{-6s}}{50s+1} \\ \frac{5.39e^{-4s}}{50s+1} & \frac{5.72e^{-2s}}{60s+1} & \frac{6.90e^{-2s}}{40s+1} \\ \frac{4.30e^{-4s}}{33s+1} & \frac{4.42e^{-4s}}{44s+1} & \frac{7.20}{19s+1} \end{bmatrix} \tag{2.6}$$

The constraints on manipulated variables are given by $|u_i(k+n)| \leq 10$, $n \geq 0$, $i = 1,2,3$. MPC is designed to control this process assuming there is no plant-model mismatch (i.e the model used by MPC is that of the process). The simulation results for set-point tracking in the controlled variables of [3,3,-3] for y1,y2,and y3; are shown respectively in Figures 2.5 and 2.6.

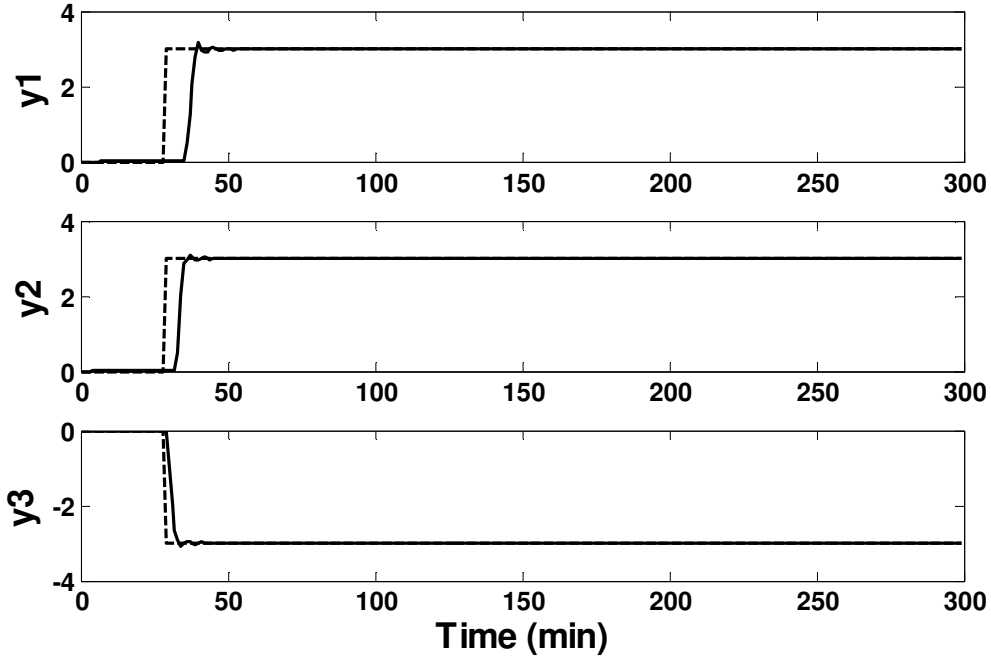


Figure 2.5 Dynamic response (no plant-model mismatch)

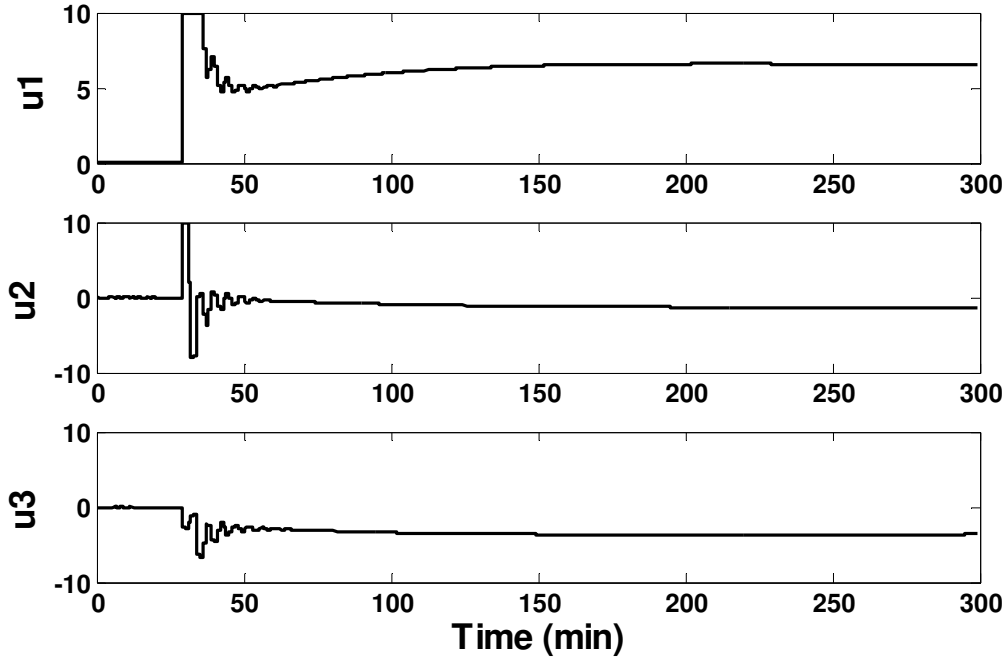


Figure 2.6 Control actions (no plant-model mismatch)

From the figures, MPC successfully tracked the given set-points providing smooth closed-loop response with feasible control actions. Now let us consider that the actual process model is given by $\boldsymbol{G}_{\text{process}} = 0.4\boldsymbol{G}_{\text{model}}$ which represents model errors in terms of steady-state gains. The simulation results are given in Figures 2.7 and 2.8. Now there is a significant offset in the responses since MPC first control action $\boldsymbol{u}_1$ saturates immediately when the set-points start to change due to plant-model mismatch. Tuning the controllers input weights could not provide any improvement. This example will be revisited later on in chapters 4 and 5 where robust DMPC algorithms are proposed.

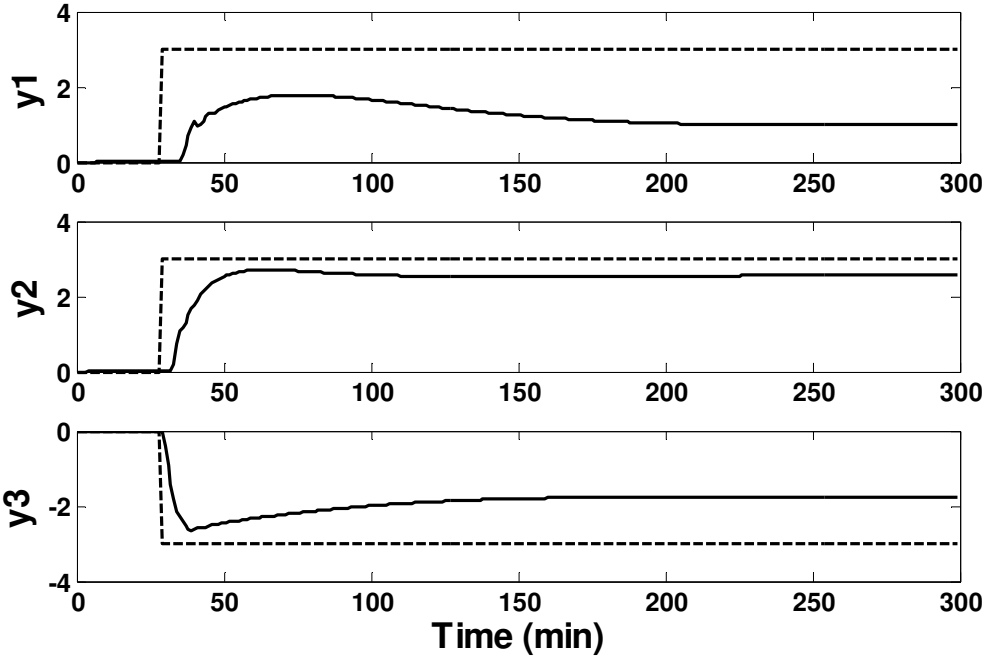


Figure 2.7 Dynamic response (with plant-model mismatch)

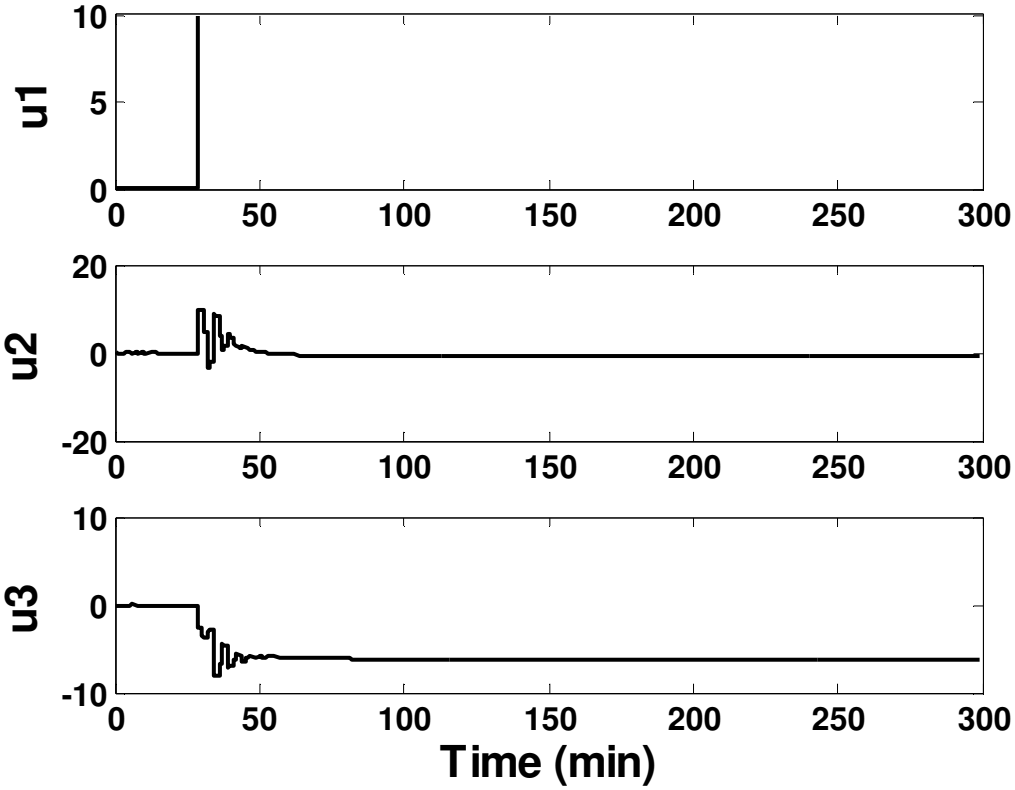


Figure 2.8 Control actions (with plant-model mismatch)

Linear Matrix Inequalities (LMI) are widely used in the design of MPC that are robust to model errors (Kothare *et al.*, 1996; Camacho and Bordons, 2003, *Bars et al.* 2006). Another attractive feature of LMIs is that they employ the efficient interior-point methods that can be solved in polynomial time. The LMI solvers are available in software such as MATLAB® which can also be integrated with YALMIP (lofberg, 2004) to formulate many convex LMI problems. A comprehensive introduction to LMIs is given by VanAntwerp and Braatz (2000). Also Boyd et al. (1994) have provided a comprehensive introduction to LMI's concepts and applications. In the remaining of this section a brief review on the application of LMI's for control design and synthesis is provided.

A linear matrix inequality can be expressed according to the following form:

$$\boldsymbol{F}(x) = \boldsymbol{F}_o + \sum_{i=1}^{m} x_i \boldsymbol{F}_i > \boldsymbol{0} \tag{2.7}$$

where $\boldsymbol{F}_i$ are symmetrical real n × n matrices, x_i are variables and $\boldsymbol{F}(x) > \boldsymbol{0}$ is positive definite. Three main problems that can be solved by LMI's are: the feasibility problem; the linear programming problem, and the generalized eigenvalue minimization problem. In addition to its ability to deal explicitly with plant model uncertainty, LMI formulation is an attractive choice for the solution of complex problems due to the availability of efficient numerical convex optimization algorithms. One of the most widely used algebraic manipulations for the formulation of LMI's is the Schur complement lemma. This lemma plays a major role in the current work and therefore it is reviewed below. Considering the following convex nonlinear inequalities:

$$R(x) > 0,\ Q(x) - S(x)R(x)^{-1}S(x)^T > 0 \tag{2.8}$$

Where $Q(x) = Q(x)^T, R(x) = R(x)^T$, and $S(x)$ depend affinely on x. The Schur complement lemma converts (2.8) into the following equivalent LMI:

$$\begin{bmatrix} Q(x) & S(x) \\ S(x)^T & R(x) \end{bmatrix} > 0 \tag{2.9}$$

The proof of Schur complement can be found in the VanAntwerp and Braatz tutorial (2000).

Kothare *et al.* (1996) proposed a formal theoretical approach for robust MPC synthesis via an online robust MPC algorithm based on LMI concepts. The algorithm guarantees robust stability as well as compliance with process constraints. The algorithm can be applied to both norm-bounded structured uncertainty descriptions and to polytopic descriptions. The latter is used in the current work and presented in the next chapters. The basic idea of their approach is that the quadratic optimization problem is converted to an LMI optimization problem that can be solved with computationally efficient interior-point algorithms (Boyd *et al.* 1994). These algorithms are very fast and can be used for online computations. LMI solvers are available in the MATLAB® Robust Control Toolbox. The following section shows typical LMIs formulations used to achieve certain design objectives.

2.5.1 stability

Considering the dynamical system:

$$\boldsymbol{\eta}(\mathrm{k}+1) = \boldsymbol{A}\,\boldsymbol{\eta}(\mathrm{k});\ \boldsymbol{\eta}(0) = \boldsymbol{\eta}_0 \tag{2.10}$$

If $\boldsymbol{A}$ is assumed to vary within a polytope of set $\{\boldsymbol{A}_1,\ldots,\boldsymbol{A}_\mathrm{k}\}$ then a sufficient condition for the asymptotic stability of this system from Lyapunov-stability theory is the feasibility of a set of LMIs as follows:

$$\text{Find } \boldsymbol{P} > 0,\ \boldsymbol{P} = \boldsymbol{P}^{\mathrm{T}} \text{ such that } \boldsymbol{A}_\mathrm{i}^{\mathrm{T}}\boldsymbol{P}\boldsymbol{A}_\mathrm{i} - \boldsymbol{P} < 0,\ \mathrm{i} = 1, \ldots, \mathrm{k} \tag{2.11}$$

2.5.2 Closed-Loop Robust Performance and the RMS Gain

For a stable state-space closed-loop system of the following form:

$$\begin{bmatrix} \boldsymbol{\eta}(k+1) \\ \boldsymbol{e}(k) \end{bmatrix} = \begin{bmatrix} \boldsymbol{A} & \boldsymbol{B} \\ \boldsymbol{C} & \boldsymbol{D} \end{bmatrix} \begin{bmatrix} \boldsymbol{\eta}(k) \\ \boldsymbol{v}(k) \end{bmatrix} \tag{2.12}$$

where $\boldsymbol{e}$(k) is the output error of the system and $\boldsymbol{v}$(k) is the input of the system (in practical applications, this can be either a set-point signal or external disturbance). The random-mean-squares (RMS) gain is the largest input/output gain γ, $\|\boldsymbol{e}\|_2 < \gamma \|\boldsymbol{v}\|_2$, over all bounded inputs. This gain is the global minimum of the following minimization problem:

$$\min_{\boldsymbol{P}\succ 0,\boldsymbol{P}=\boldsymbol{P}^T} \gamma^2 \tag{2.13}$$

$$\text{Subject to} \begin{bmatrix} \boldsymbol{A}^T\boldsymbol{P}\boldsymbol{A}-\boldsymbol{P} & \boldsymbol{A}^T\boldsymbol{P}\boldsymbol{B} & \boldsymbol{C}^T \\ \boldsymbol{B}^T\boldsymbol{P}\boldsymbol{A} & \boldsymbol{B}^T\boldsymbol{P}\boldsymbol{B}-\gamma^2\boldsymbol{I} & \boldsymbol{D}^T \\ \boldsymbol{C} & \boldsymbol{D} & -\boldsymbol{I} \end{bmatrix} < \boldsymbol{0}$$

The optimization problem in (2.13) will play a major role in Chapter 3 since the index γ is used in the current research to select the best control structure in the context of distributed MPC.

2.6 Explicit Linear Model Predictive Control

In general, one of the major drawbacks of MPC is that it is computationally expensive for online implementation especially when constraints are considered in the optimization problem. These computational requirements grow exponentially as the problem size increases. Consequently, reducing online computational burdens is of key importance for real time implementation. To reduce computations a multiparametric programming approach has been proposed where an explicit control law solution is used thus convert all or most of online computations into offline ones (Pistikopoulos *et al.* 2007). The explicit solution is then used in an MPC online implementation. The control moves become an affine function of the state variables so the online implementation requires searching through a look-up table. The offline framework is composed of three steps: (1) solving the optimization problem offline using multiparametric programming

approach, (2) partitioning the state space, and (3) obtaining the optimal control moves online.

In the last few years a significant number of papers have been published dealing with the development of multiparametric or explicit MPC algorithms. A brief review is presented in the next paragraphs.

Bemporad *et al.* (2002) developed an algorithm for explicit linear MPC based on a multiparametric programming approach. The state-feedback solution was obtained for nominal MPC. The original online MPC problem was converted to a multiparametric quadratic programming (mp-QP) that can be solved offline and the optimal inputs were obtained as affine functions of the states. Partitioning the state space into a number of convex polyhedral regions was also described. This number depends on many parameters such as the dimension of the state vector, the number of control moves, and the number of constraints. The algorithm assumed perfect model and therefore uncertainty in the model was not accounted for.

Wan and Kothare (2003) used the concept of an asymptotically stable invariant ellipsoid to develop an explicit robust MPC algorithm that provides a sequence of explicit solutions obtained offline. The algorithm was based on the original framework reported in Kothare *et al.* (1996).

In (Chu 2006; Chu *et al.* 2006) an algorithm was developed to obtain robust MPC explicit solutions for constrained multivariable problems with internal and external uncertainties.

In general, all the explicit MPC techniques suffer from a common problem which is the computational complexity that grows dramatically with the problem size (Pistikopoulos *et al.* 2007). However, attempts are in progress to overcome this problem.

2.7 Main Assumptions

The key assumptions made throughout this work are summarized as follows:

1) The system is controllable and observable.
2) The system can be decomposed into smaller subsystems that are controllable and observable.
3) The model uncertainty can be represented by a set of linear state-space models.
4) The performance metric is given as a quadratic objective function that can be decomposed.
5) Reliable communication network is available to exchange information.
6) The entire vector of states measurements or estimates is made available to all controllers to guarantee overall system's stability.
7) Control intervals are the same in all controllers.

CHAPTER 3

Selection of Control Structure for Distributed Model Predictive Control in the Presence of Model Errors

Adapted from Al-Gherwi et al. (2010)

3.1 Overview

This chapter presents a new methodology for selecting control structure in the context of distributed model predictive control. An index was developed to quantify the performance of distributed MPC strategies in the presence of model errors. This index was used for two purposes: to solve the decomposition problem whereby the process is decomposed into parts and to compare distributed MPC strategies with different degrees of coordination. Then, a multi-objective Mixed Integer Nonlinear Programming MINLP formulation is proposed to achieve an optimal tradeoff between performance and structure simplicity.

Four examples are considered to illustrate the methodology. The simulation results are consistent with the conclusions obtained from the analysis. The proposed methodology can be applied at the design stage to select the best control configuration in the presence of model errors.

3.2 Introduction

Since the advent of model predictive control (MPC) technology, the process industry has witnessed a gradual shift from the conventional multi-loop decentralized PID control strategies to centralized multivariable MPC control. The ability of MPC to

handle process constraints and interactions among process units attracted the practitioners to implement MPC (Qin and Badgwell 2003). However, using one centralized MPC strategy has some drawbacks related to high computational demand especially in processes with relatively large number of inputs and outputs, to sensitivity to model errors and to low resilience with respect to operational changes (Skogestad 2004; Skogestad and Postlethwaite 2005). This led to the idea of partitioning the original process into smaller units or subsystems and applying MPC controllers to each one of these subsystems. The simultaneous operation of several MPC controllers in such fashion has been referred to in the literature as Distributed MPC. When the individual MPC controllers for the different subsystems are operated in a completely decentralized fashion, closed loop performance may be significantly hampered since some or all of the interactions are ignored and also the controllers may become unstable if these interactions are strong. As a remedy to this problem, researchers have proposed the use of some form of coordination between the MPC controllers for the different subsystems. The main idea is to decompose the centralized dynamic model of the system into local models for each subsystem while interaction models are used to filter the communication of relevant information between the subsystems (Rawlings and Stewart 2008). While all the reported methods share the same idea of communication to account for interaction, the major difference is in the type of local objective function to be solved by every controller and in the way that their operation is coordinated in order to achieve global objectives. On one hand, when the local objective of each controller does not account for the goals of other controllers then two types of strategies arise: a decentralized strategy results if the interactions are ignored or a Nash-equilibrium based strategy results if the interactions

are accounted for (Li *et al.* 2005; Mercangöz and Doyle 2007). On the other hand, when each local objective is modified to take into account the goals of other controllers, strategies referred to as feasible-cooperative were proposed that seek to optimize a combination of all related objectives (Venkat 2006; Zhang *et al.* 2007). Figure 3.1 illustrates the general structure and information exchange for different possible MPC strategies: centralized, fully decentralized, and coordination-based. The later include both Nash equilibrium based strategies as well as cooperative strategies that minimize an overall objective function.

In centralized MPC the optimal control moves are obtained by minimizing a cost function that takes into account the overall objective of the entire system. Since the centralized control considers the interactions within the system, optimal nominal performance is expected. In contrast, for distributed MPC, either fully decentralized or coordinated, the cost functions are local and the interactions are either completely ignored or they are accounted for in a partial manner. Thus, distributed MPC strategies have a simpler control structure but their performance is expected to be generally poorer as compared to centralized MPC strategies. Thus there is a trade-off between the best achievable closed-loop performance and the simplicity of the controller structure. In a recent review on distributed MPC literature Scattolini has identified the importance of addressing and developing tools to search for such a trade-off (Scattolini 2009).

Beyond the controller structure selection problem that involves assigning the inputs and outputs to the different subsystems, the application of the special case of fully decentralized MPC also requires assigning parts of the states' vector to the subsystems. This has been referred to in the literature as the *decomposition problem* but it has not

been systematically solved in the context of distributed MPC problem. Decomposition can be obtained based on open-loop information but this may lead to conservative stability results (Samyudia *et al.* 1994) However, the particular question addressed in the current study is on the level of performance that can be specifically achieved with decentralized MPC by changing tuning parameters.

In addition it has been recognized that the performance of multivariable controllers is highly dependent on their robustness to model errors. Most distributed MPC formulations reported in the literature use linear models and rely on feedback to account for uncertainty. However, in reality linear models are never accurate due to nonlinearity or inaccurate identification. Although the study of robustness has been identified as a key factor for the successful application of distributed MPC, this problem has not been systematically analyzed in the context of distributed MPC other than by simulations (Rawlings and Stewart 2008).

Following the above, the current work will address the following goals:

i- An index will be developed to quantify the performance of distributed MPC strategies in the presence of model errors. This index will be used for two purposes: to solve the decomposition problem and to compare distributed MPC strategies with different degrees of coordination.

ii- A multi-objective Mixed Integer Nonlinear Programming MINLP formulation is proposed for seeking an optimal tradeoff between performance and structure simplicity. The later is quantified by an index that is proportional to the number of interactions included in the control strategy.

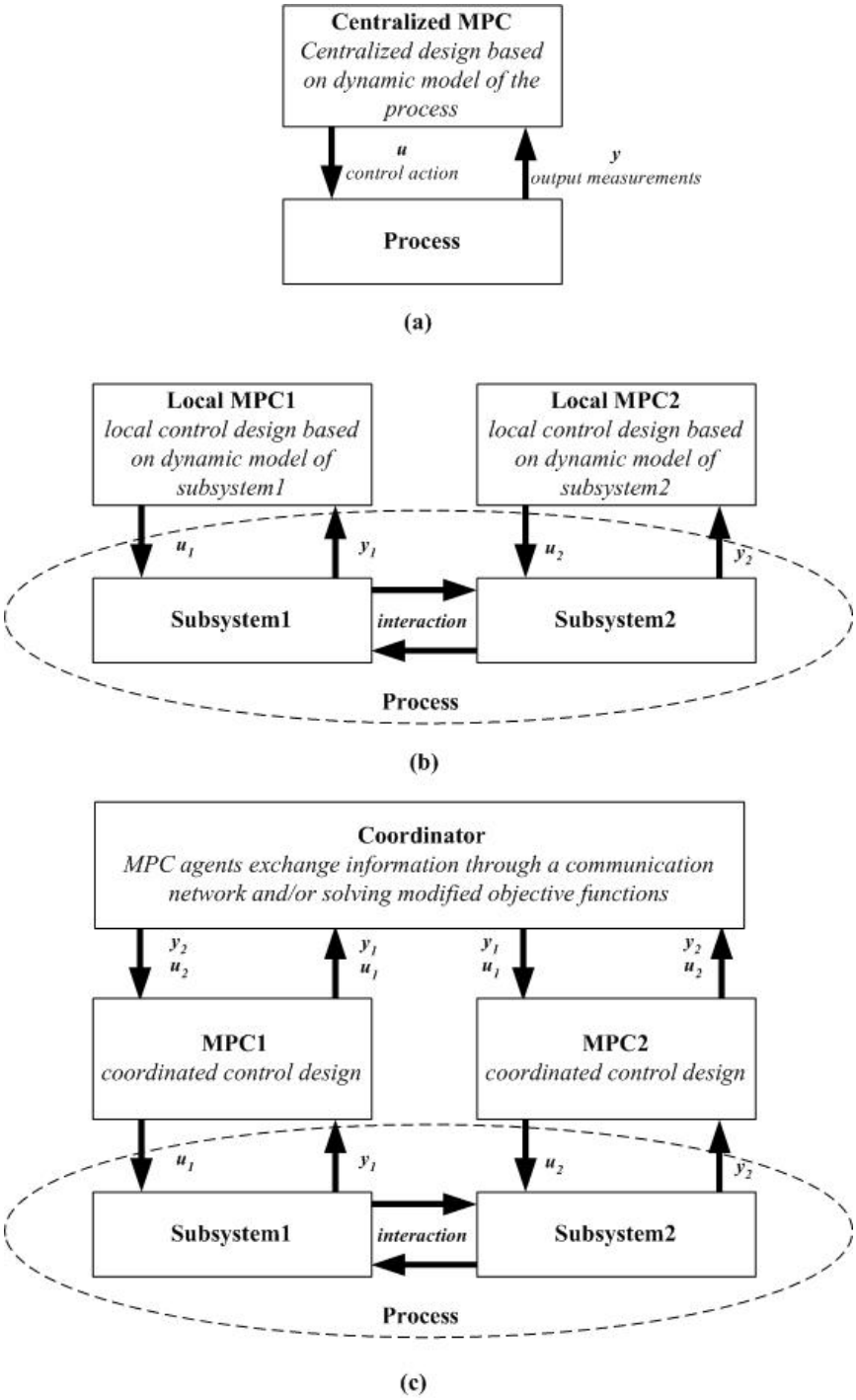


Figure 3.1 Information structure: (a) Centralized MPC, (b) Fully decentralized MPC, and (c) Coordinated MPCs.

This chapter is organized as follows. In section 3.3 basic definitions and the methodology are shown. Then the application of the methodology to four case studies is shown in section 3.4. Conclusions are presented in Section 3.5.

3.3 **Definitions and Methodology**

3.3.1 **Models**

The nominal model of the process used by the MPC is given by the following discrete linear time-invariant (LTI) state-space model:

$$\boldsymbol{x}(k+1) = \boldsymbol{A}\boldsymbol{x}(k) + \boldsymbol{B}\boldsymbol{u}(k) \tag{3.1}$$

$$\boldsymbol{y}(k) = \boldsymbol{C}\boldsymbol{x}(k) \tag{3.2}$$

where $\boldsymbol{x}(k) \in \Re^{nx}$ is an nx-dimensional state vector; $\boldsymbol{u}(k) \in \Re^{nu}$ is an nu-dimensional input vector; $\boldsymbol{y}(k) \in \Re^{ny}$ is an ny-dimensional output vector; $\boldsymbol{A} \in \Re^{nx \times nx}$ is the state matrix, $\boldsymbol{B} \in \Re^{nx \times nu}$ is the input matrix, and $\boldsymbol{C} \in \Re^{ny \times nx}$ is the measurement matrix; k is the time interval. It is assumed that the sets ($\boldsymbol{A}$,$\boldsymbol{B}$) and ($\boldsymbol{A}$,$\boldsymbol{C}$) are controllable and observable; respectively.

It is assumed that the actual process to be controlled is represented by the following linear time-varying (LTV) model:

$$\boldsymbol{x}_p(k+1) = \boldsymbol{A}_p(k)\boldsymbol{x}_p(k) + \boldsymbol{B}_{pu}(k)\boldsymbol{u}(k) + \boldsymbol{B}_{pw}(k)\boldsymbol{w}(k) \tag{3.3}$$

$$\boldsymbol{y}_p(k) = \boldsymbol{C}_p\boldsymbol{x}_p(k) \tag{3.4}$$

where $\boldsymbol{w}(k) \in \Re^{nw}$ is an *nw*-dimensional exogenous input disturbance vector. In both models (3.1)-(3.2) and (3.3)-(3.4) it is assumed that the number of states is the same, however, the parameters are not necessary the same. For the set-point tracking problem *nw* is set to 0. The disturbance model is assumed to be given by the following equation:

$$\boldsymbol{w}(k+1) = \alpha_w \boldsymbol{w}(k) + (1 - \alpha_w)\boldsymbol{d}(k) \tag{3.5}$$

where $\boldsymbol{d}(k)$ is an unmeasured white noise entering the system and, $1 \leq \alpha_w \leq 0$ is a first order filter constant necessary to limit the bandwidth of this disturbance for performance analysis purposes. Due to the presence of model errors, the plant model given in (3.3)-(3.4) is represented by a convex set $\boldsymbol{\pi}$ of linear plants with *L* vertices defined as (Boyd *et al.* 1994):

$$\boldsymbol{\pi}=[\boldsymbol{A}_p(k)\ \boldsymbol{B}_{pu}(k)\ \boldsymbol{B}_{pw}(k)] = \sum_{i=1}^{L} \xi_i [\boldsymbol{A}_{pi}\ \boldsymbol{B}_{pui}\ \boldsymbol{B}_{pwi}] \quad ;\forall \xi_i \geq 0, \sum_{i=1}^{L} \xi_i = 1 \tag{3.6}$$

Accordingly, it is assumed that the actual plant model lies within a polytope of matrices as defined above. The vertices of this polytope correspond to the extreme values of a family of linear models obtained from the linearization of the nonlinear system model or alternatively from identification around different operating points. It is assumed that all the plants have the same number of states to satisfy the above definition.

3.3.2 MPC Strategies

Three MPC strategies are considered: a centralized control strategy and two distributed MPC strategies, one designed as Nash-based distributed MPC and another designed as a fully decentralized MPC. The feasible-cooperative type strategy mentioned in the introduction section has not been considered in this study since it is expected to perform similarly to centralized MPC provided that convergence is reached (Venkat 2006; Zhang *et al.* 2007). A description of the structure of the used strategies follows in order.

In the present study, the centralized MPC is based on the formulation given in (Maciejowski 2002). The cost function is defined as:

$$\min_{\Delta U} J(k) = \left\| \boldsymbol{Y}(k|k) - \boldsymbol{T}(k) \right\|_{\boldsymbol{Q}}^2 + \left\| \Delta \boldsymbol{U}(k|k) \right\|_{\boldsymbol{\Lambda}}^2 \tag{3.7}$$

where $\boldsymbol{Y}(k|k) = [\boldsymbol{y}(k+1|k)^T,\ldots,\boldsymbol{y}(k+Hp|k)^T]^T$ is the vector of predicted outputs; Hp is the prediction horizon; $\boldsymbol{T}(k) = [\boldsymbol{R}(k+1)^T,\ldots,\boldsymbol{R}(k+Hp)^T]^T$ is the vector of set-points; $\Delta\boldsymbol{U}(k|k) = [\Delta\boldsymbol{u}(k/k)^T,\ldots,\Delta\boldsymbol{u}(k+Hu-1/k)^T]^T$; Hu is the control horizon; $\Delta\boldsymbol{u}(k/k) = \boldsymbol{u}(k/k)-\boldsymbol{u}(k-1/k-1)$; $\boldsymbol{Q} = block\text{-}diag(\boldsymbol{Q}_1,\ldots,\boldsymbol{Q}_{Hp})$ and $\boldsymbol{\Lambda} = block\text{-}diag(\boldsymbol{\lambda}_1,\ldots,\boldsymbol{\lambda}_{Hu})$ are the output weights matrix and the input weights matrix; respectively. The weighted vector norms are defined as $\|\boldsymbol{v}\|_A^2 = \boldsymbol{v}^T\boldsymbol{A}\boldsymbol{v}$. The set-point signal $\boldsymbol{R}(k)$ is obtained by filtering the original set-point signal $\boldsymbol{r}(k) = [\boldsymbol{r}_1(k),\ldots,\boldsymbol{r}_{ny}(k)]^T$ according to the following first order exponential filter:

$$\boldsymbol{R}(k+1) = \alpha\boldsymbol{R}(k) + (1-\alpha)\boldsymbol{r}(k) \tag{3.8}$$

where $1 \leq \alpha \leq 0$ is the filter parameter that is specified by the user based on the desired set-point bandwidth and r is assumed to be white noise. This filter is equivalent to the robustness filter used in internal model control (IMC) in order to shape the response of the closed-loop system (Ricker 1990). The set-point $\boldsymbol{r}(k)$ is set to 0 in the disturbance rejection problem.

From the nominal model (3.1)-(3.2), the predicted output vector is obtained as follows:

$$\begin{bmatrix} \boldsymbol{y}(k+1|k) \\ \boldsymbol{y}(k+2|k) \\ \vdots \\ \boldsymbol{y}(k+Hp|k) \end{bmatrix} = \begin{bmatrix} \boldsymbol{CA} \\ \boldsymbol{CA}^2 \\ \vdots \\ \boldsymbol{CA}^{Hp} \end{bmatrix} \boldsymbol{x}(k) + \begin{bmatrix} \boldsymbol{CB} \\ \boldsymbol{C}(\boldsymbol{AB}+\boldsymbol{B}) \\ \vdots \\ \sum_{i=0}^{Hp-1} \boldsymbol{CA}^i \boldsymbol{B} \end{bmatrix} \boldsymbol{u}(k-1) + \begin{bmatrix} \boldsymbol{CB} & \cdots & \boldsymbol{0} \\ \boldsymbol{C}(\boldsymbol{AB}+\boldsymbol{B}) & \cdots & \boldsymbol{0} \\ \vdots & \ddots & \vdots \\ \sum_{i=0}^{Hp-1} \boldsymbol{CA}^i \boldsymbol{B} & \cdots & \sum_{i=0}^{Hp-Hu} \boldsymbol{CA}^i \boldsymbol{B} \end{bmatrix} \begin{bmatrix} \Delta \boldsymbol{u}(k|k) \\ \vdots \\ \Delta \boldsymbol{u}(k+Hu-1|k) \end{bmatrix} + \boldsymbol{\Xi}(k|k) \tag{3.9}$$

where the term $\boldsymbol{\Xi}(k|k)$ accounts for unmeasured disturbances and/or model errors due to the difference between the nominal model in (3.1)-(3.2) and the plant model in (3.3)-(3.4). Following the assumption commonly used in the classical dynamic matrix control DMC algorithms (Cutler and Ramaker 1979), the elements in $\boldsymbol{\Xi}(k|k)$ are assumed to remain constant along the horizon of Hp time intervals. Thus, $\boldsymbol{\Xi}(k|k)$ is defined as follows:

$$\boldsymbol{\Xi}(k|k) = \boldsymbol{Lc}\left[\boldsymbol{y}_p(k) - \boldsymbol{y}(k|k-1)\right] \tag{3.10}$$

$\boldsymbol{Lc} = \left[\boldsymbol{I}_{ny}, \cdots, \boldsymbol{I}_{ny}\right]^T_{ny \times Hp}$, where $\boldsymbol{I}_{ny}$ is the identity matrix.

The prediction expression given in (3.9) can be rewritten in the following compact form:

$$\boldsymbol{Y}(k|k) = \boldsymbol{\Psi x}(k) + \boldsymbol{\Gamma u}(k-1) + \boldsymbol{\Theta \Delta U}(k) + \boldsymbol{\Xi}(k|k) \tag{3.11}$$

The tracking error vector of the free-response $\boldsymbol{E}(k|k)$ is defined as:

$$\boldsymbol{E}(k|k) = \boldsymbol{T}(k) - \boldsymbol{\Psi x}(k) - \boldsymbol{\Gamma u}(k-1) - \boldsymbol{\Xi}(k|k) \tag{3.12}$$

The optimal moves at the current step (k), $\boldsymbol{\Delta u}^*(k|k)$, are obtained by solving (3.7) subject to (3.9) in the absence of constraints and given as:

$$\boldsymbol{\Delta u}^*(k|k) = \boldsymbol{K}_{MPC}\boldsymbol{E}(k|k) \tag{3.13}$$

$$\boldsymbol{K}_{MPC} = [\boldsymbol{I}_{nu}, \boldsymbol{0}_{nu}, \ldots, \boldsymbol{0}_{nu}]_{nu \times Hu}(\boldsymbol{\Theta}^T\boldsymbol{Q}\boldsymbol{\Theta} + \boldsymbol{\Lambda})^{-1}\boldsymbol{\Theta}^T\boldsymbol{Q}$$

At this point, the closed-loop system for every model included in the set $\boldsymbol{\pi}$ defined in (3.6) with the centralized MPC can be easily obtained for both set-point tracking and disturbance rejection problems. To simplify the notations, it is assumed without loss of

generality that $\boldsymbol{u}(k) = \boldsymbol{u}(k|k)$ and $\boldsymbol{u}(k\text{-}1) = \boldsymbol{u}(k\text{-}1|k\text{-}1)$. The resulting closed-loop system for both problems is given below.

Set-point tracking problem

$$\begin{bmatrix} \boldsymbol{\eta}_r(k+1) \\ \boldsymbol{e}_r(k) \end{bmatrix} = \begin{bmatrix} \boldsymbol{A}_{CL}(k) & \boldsymbol{B}_{CL} \\ \boldsymbol{C}_{CL} & \boldsymbol{D}_{CL} \end{bmatrix} \begin{bmatrix} \boldsymbol{\eta}_r(k) \\ \boldsymbol{r}(k) \end{bmatrix}; \forall \boldsymbol{A}_{CL}(k) \in Co\left\{\boldsymbol{A}_{CL_1}, \cdots \boldsymbol{A}_{CL_L}\right\} \tag{3.14}$$

where $\boldsymbol{\eta}_r(k) = [\boldsymbol{x}_p(k)^T, \boldsymbol{x}(k)^T, \boldsymbol{u}(k\text{-}1)^T, \boldsymbol{R}(k)^T]^T$; the vertex $\boldsymbol{A}_{CL_i}$ in the above convex hull with the rest of closed-loop matrices are given as follows:

$$\boldsymbol{A}_{CL_i} = \begin{bmatrix} \boldsymbol{A}_{pi} - \boldsymbol{B}_{pui}\boldsymbol{K}_{MPC}\boldsymbol{L}\boldsymbol{c}\boldsymbol{C}_p & \boldsymbol{B}_{pui}\boldsymbol{K}_{MPC}\left(\boldsymbol{L}\boldsymbol{c}\boldsymbol{C} - \boldsymbol{\Psi}\right) \\ -\boldsymbol{B}\boldsymbol{K}_{MPC}\boldsymbol{L}\boldsymbol{c}\boldsymbol{C}_p & \boldsymbol{A} + \boldsymbol{B}\boldsymbol{K}_{MPC}\left(\boldsymbol{L}\boldsymbol{c}\boldsymbol{C} - \boldsymbol{\Psi}\right) \\ -\boldsymbol{K}_{MPC}\boldsymbol{L}\boldsymbol{c}\boldsymbol{C}_p & \boldsymbol{K}_{MPC}\left(\boldsymbol{L}\boldsymbol{c}\boldsymbol{C} - \boldsymbol{\Psi}\right) \\ \boldsymbol{0}_{ny\times nx} & \boldsymbol{0}_{ny\times nx} \end{bmatrix}$$

$$\begin{bmatrix} \boldsymbol{B}_{pui} - \boldsymbol{B}_{pui}\boldsymbol{K}_{MPC}\boldsymbol{\Gamma} & \boldsymbol{B}_{pui}\boldsymbol{K}_{MPC}\boldsymbol{L}\boldsymbol{c} \\ \boldsymbol{B} - \boldsymbol{B}\boldsymbol{K}_{MPC}\boldsymbol{\Gamma} & \boldsymbol{B}\boldsymbol{K}_{MPC}\boldsymbol{L}\boldsymbol{c} \\ \boldsymbol{I}_{nu} - \boldsymbol{K}_{MPC}\boldsymbol{\Gamma} & \boldsymbol{K}_{MPC}\boldsymbol{L}\boldsymbol{c} \\ \boldsymbol{0}_{ny\times nu} & \alpha\boldsymbol{I}_{ny} \end{bmatrix}$$

$$\boldsymbol{B}_{CL} = \begin{bmatrix} \boldsymbol{0}_{nx\times ny} \\ \boldsymbol{0}_{nx\times ny} \\ \boldsymbol{0}_{nu\times ny} \\ (1-\alpha)\boldsymbol{I}_{ny} \end{bmatrix};\ \boldsymbol{C}_{CL} = \begin{bmatrix} -\boldsymbol{C}_p \\ \boldsymbol{0}_{ny\times nx} \\ \boldsymbol{0}_{ny\times nu} \\ \boldsymbol{I}_{ny} \end{bmatrix}^T;\ \boldsymbol{D}_{CL} = \begin{bmatrix} \boldsymbol{0}_{ny\times ny} \end{bmatrix} \tag{3.15}$$

and the output error signal is defined as $\boldsymbol{e}_r(k) = \boldsymbol{R}(k) - \boldsymbol{y}_p(k)$.

Disturbance rejection problem

$$\begin{bmatrix} \boldsymbol{\eta}_d(k+1) \\ \boldsymbol{e}_d(k) \end{bmatrix} = \begin{bmatrix} \boldsymbol{A}_{CL}(k) & \boldsymbol{B}_{CL} \\ \boldsymbol{C}_{CL} & \boldsymbol{D}_{CL} \end{bmatrix} \begin{bmatrix} \boldsymbol{\eta}_r(k) \\ \boldsymbol{d}(k) \end{bmatrix}; \forall \boldsymbol{A}_{CL}(k) \in Co\left\{\boldsymbol{A}_{CL_1}, \cdots \boldsymbol{A}_{CL_L}\right\} \tag{3.16}$$

where $\boldsymbol{\eta}_d(k) = [\boldsymbol{x}_p(k)^T, \boldsymbol{x}(k)^T, \boldsymbol{u}(k-1)^T, \boldsymbol{w}(k)^T]^T$; the vertex $\boldsymbol{A}_{CL_i}$ in the above convex hull with the rest of closed-loop matrices are given as follows:

$$\boldsymbol{A}_{CL_i} = \begin{bmatrix} \boldsymbol{A}_{pi} - \boldsymbol{B}_{pui}\boldsymbol{K}_{MPC}\boldsymbol{L}\boldsymbol{c}\boldsymbol{C}_p & \boldsymbol{B}_{pui}\boldsymbol{K}_{MPC}\left(\boldsymbol{L}\boldsymbol{c}\boldsymbol{C} - \boldsymbol{\Psi}\right) & \boldsymbol{B}_{pui} - \boldsymbol{B}_{pui}\boldsymbol{K}_{MPC}\boldsymbol{\Gamma} & \boldsymbol{B}_{pwi} \\ -\boldsymbol{B}\boldsymbol{K}_{MPC}\boldsymbol{L}\boldsymbol{c}\boldsymbol{C}_p & \boldsymbol{A} + \boldsymbol{B}\boldsymbol{K}_{MPC}\left(\boldsymbol{L}\boldsymbol{c}\boldsymbol{C} - \boldsymbol{\Psi}\right) & \boldsymbol{B} - \boldsymbol{B}\boldsymbol{K}_{MPC}\boldsymbol{\Gamma} & \boldsymbol{0}_{nx\times nw} \\ -\boldsymbol{K}_{MPC}\boldsymbol{L}\boldsymbol{c}\boldsymbol{C}_p & \boldsymbol{K}_{MPC}\left(\boldsymbol{L}\boldsymbol{c}\boldsymbol{C} - \boldsymbol{\Psi}\right) & \boldsymbol{I}_{nu} - \boldsymbol{K}_{MPC}\boldsymbol{\Gamma} & \boldsymbol{0}_{nu\times nw} \\ \boldsymbol{0}_{nw\times nx} & \boldsymbol{0}_{nw\times nx} & \boldsymbol{0}_{nw\times nu} & \alpha_w \boldsymbol{I}_{nw} \end{bmatrix};$$

$$\boldsymbol{B}_{CL} = \begin{bmatrix} \boldsymbol{0}_{nx\times nw} \\ \boldsymbol{0}_{nx\times nw} \\ \boldsymbol{0}_{nu\times nw} \\ (1-\alpha_w)\boldsymbol{I}_{nw} \end{bmatrix}; \ \boldsymbol{C}_{CL} = \begin{bmatrix} -\boldsymbol{C}_p \\ \boldsymbol{0}_{ny\times nx} \\ \boldsymbol{0}_{ny\times nu} \\ \boldsymbol{0}_{ny\times nw} \end{bmatrix}^T; \ \boldsymbol{D}_{CL} = \begin{bmatrix} \boldsymbol{0}_{ny\times nw} \end{bmatrix} \tag{3.17}$$

and the output error signal is defined as $\boldsymbol{e}_d(k) = -\boldsymbol{y}_p(k)$.

For distributed MPC, the centralized nominal model given in (3.1)-(3.2) is decomposed into N subsystems where the model of subsystem $i \in \{1,.....,N\}$ can be written as:

$$\boldsymbol{x}_i(k+1) = \boldsymbol{A}_i\boldsymbol{x}_i(k) + \boldsymbol{B}_{ii}\boldsymbol{u}_i(k) + \sum_{\substack{j=1 \\ j\neq i}}^{N} \delta_{ij}\boldsymbol{B}_{ij}\boldsymbol{u}_j \tag{3.18}$$

$$\boldsymbol{y}_i(k) = \boldsymbol{C}_i\boldsymbol{x}_i(k) \tag{3.19}$$

where $\boldsymbol{x}_i(k) \in \Re^{nx_i}$ is an nx_i-dimensional augmented state vector of the subsystem i defined as $\boldsymbol{x}_i^T = \left[\boldsymbol{x}_{11}^T, \cdots, \boldsymbol{x}_{ii}^T, \cdots, \boldsymbol{x}_{NN}^T\right]^T$ with states $\boldsymbol{x}_{ii}$ that are measured locally within subsystem i and the other states $\boldsymbol{x}_{ij}$ that are affecting subsystem i are measured within the j^{th} subsystem and can be exchanged via communication; $\boldsymbol{u}_i \in \Re^{nu_i}$ is an nu_i-dimensional input vector estimated by the i^{th} MPC assigned to subsystem i; $\boldsymbol{u}_j \in \Re^{nu_j}$ is an nu_j-dimensional input vector estimated by the j^{th} MPC assigned to subsystem j and affects subsystem i; $\boldsymbol{y}_i \in \Re^{ny_i}$ is an ny_i-dimensional output vector; δ_{ij} is a switching variable, either 0 or 1, and is used in the proposed methodology to either neglect or include the corresponding interaction term as explained later. $\boldsymbol{A}_i$, $\boldsymbol{B}_{ii}$, $\boldsymbol{B}_{ij}$, and $\boldsymbol{C}_i$ are matrices of appropriate dimensions and the matrix $\boldsymbol{A}_i$ is given as follows:

$$\boldsymbol{A}_i = \begin{bmatrix} \delta_{i1}\boldsymbol{A}_{11} & \cdots & \delta_{i1}\boldsymbol{A}_{1i} & \cdots & \delta_{i1}\boldsymbol{A}_{1N} \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ \delta_{i1}\boldsymbol{A}_{i1} & \cdots & \boldsymbol{A}_{ii} & \cdots & \delta_{iN}\boldsymbol{A}_{iN} \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ \delta_{iN}\boldsymbol{A}_{N1} & \cdots & \delta_{iN}\boldsymbol{A}_{Ni} & \cdots & \delta_{iN}\boldsymbol{A}_{NN} \end{bmatrix} \tag{3.20}$$

From the above definitions, the model in (3.18)-(3.19) can also be used to represent the special case of a fully decentralized system where all the interaction terms are ignored and this is obtained by simply setting all the corresponding δ_{ij}'s to 0 and keeping only local states and inputs of each subsystem, i.e. $\boldsymbol{A}_i = \boldsymbol{A}_{ii}$ and the effects of all $\boldsymbol{u}_j$'s, for j not equal i, are neglected. For analysis purposes, all equations for the individual subsystems are grouped together to formulate an overall model as follows:

$$\boldsymbol{x}(k+1) = \boldsymbol{A}_o \boldsymbol{x}(k) + \boldsymbol{B}_o \boldsymbol{u}(k) \tag{3.21}$$

$$\boldsymbol{y}(k) = \boldsymbol{C}_o \boldsymbol{x}(k) \tag{3.22}$$

where

$$\boldsymbol{A}_o = block\text{-}diag\,(\boldsymbol{A}_1,\ldots,\boldsymbol{A}_N);\ \boldsymbol{B}_o = \begin{bmatrix} \boldsymbol{B}_{11} & \cdots & \delta_{1N}\boldsymbol{B}_{1N} \\ \vdots & \ddots & \vdots \\ \delta_{1N}\boldsymbol{B}_{N1} & \cdots & \boldsymbol{B}_{NN} \end{bmatrix};\ \boldsymbol{C}_o = block\text{-}diag\,(\boldsymbol{C}_1,\ldots,\boldsymbol{C}_N) \tag{3.23}$$

The state vector $\boldsymbol{x}$ and the input vector $\boldsymbol{u}$ are obtained by appending all the state vectors and the input vectors of the i^{th} subsystems; respectively. Accordingly, the model given in (3.21)-(3.22) is a non-minimal realization of the system. To formulate the closed-loop systems of distributed MPC, the matrices $\boldsymbol{A}$, $\boldsymbol{B}$, and $\boldsymbol{C}$ in (3.14)-(3.15) and (3.16)-(3.17) are replaced by $\boldsymbol{A}_o$, $\boldsymbol{B}_o$, and $\boldsymbol{C}_o$; respectively.

The formulation of the Nash-based distributed MPC strategy is based on the work reported in (Li *et al.* 2005). However, since the formulation in that study was based on input/output models which are not suitable for the robustness analysis to be conducted in the current work, a formulation of the Nash-equilibrium strategy based on state-space models is presented instead. In the Nash-based MPC, the $\boldsymbol{\Delta U}_i$ manipulated variable action is calculated by minimizing the local cost function of the i^{th} subsystem as follows:

$$\min_{\boldsymbol{\Delta U}_i} J_i(k) = \left\| \boldsymbol{Y}_i(k|k) - \boldsymbol{T}_i(k) \right\|_{\boldsymbol{Q}_i}^2 + \left\| \boldsymbol{\Delta U}_i(k|k) \right\|_{\lambda_i}^2 \tag{3.24}$$

The predicted output vector of subsystem i is easily obtained by solving (3.18)-(3.19) recursively and given as:

$$\begin{aligned} \boldsymbol{Y}_i\left(k\middle|k\right) &= \boldsymbol{\Psi}_i \boldsymbol{x}_i(k) + \boldsymbol{\Gamma}_i \boldsymbol{u}_i(k-1) + \boldsymbol{\Theta}_i \boldsymbol{\Delta U}_i(k\middle|k) \\ &\quad + \sum_{j\neq i} \boldsymbol{\Gamma}_{ij} \boldsymbol{u}_j(k-1) + \sum_{j\neq i} \boldsymbol{\Theta}_{ij} \boldsymbol{\Delta U}_j(k\middle|k) + \boldsymbol{\Xi}_i(k\middle|k) \end{aligned} \tag{3.25}$$

Where

$$\boldsymbol{\Psi}_i = \begin{bmatrix} \boldsymbol{C}_i \boldsymbol{A}_i \\ \boldsymbol{C}_i \boldsymbol{A}_i^{\,2} \\ \vdots \\ \boldsymbol{C}_i \boldsymbol{A}_i^{\,Hp} \end{bmatrix};\ \boldsymbol{\Gamma}_i = \begin{bmatrix} \boldsymbol{C}_i \boldsymbol{B}_{ii} \\ \boldsymbol{C}_i\left(\boldsymbol{A}_i \boldsymbol{B}_{ii} + \boldsymbol{B}_{ii}\right) \\ \vdots \\ \sum_{k=0}^{Hp-1} \boldsymbol{C}_i \boldsymbol{A}_i^{k} \boldsymbol{B}_{ii} \end{bmatrix};\ \boldsymbol{\Theta}_i = \begin{bmatrix} \boldsymbol{C}_i \boldsymbol{B}_{ii} & \cdots & \boldsymbol{0} \\ \boldsymbol{C}_i\left(\boldsymbol{A}_i \boldsymbol{B}_{ii} + \boldsymbol{B}_{ii}\right) & \cdots & \boldsymbol{0} \\ \vdots & \ddots & \vdots \\ \sum_{k=0}^{Hp-1} \boldsymbol{C}_i \boldsymbol{A}_i^{k} \boldsymbol{B}_{ii} & \cdots & \sum_{k=0}^{Hp-Hu} \boldsymbol{C}_i \boldsymbol{A}_i^{k} \boldsymbol{B}_{ii} \end{bmatrix};$$

$$\boldsymbol{\Gamma}_{ij} = \begin{bmatrix} \boldsymbol{C}_i \delta_{ij} \boldsymbol{B}_{ij} \\ \boldsymbol{C}_i \delta_{ij}\left(\boldsymbol{A}_i \boldsymbol{B}_{ij} + \boldsymbol{B}_{ij}\right) \\ \vdots \\ \sum_{k=0}^{Hp-1} \boldsymbol{C}_i \boldsymbol{A}_i^{k} \delta_{ij} \boldsymbol{B}_{ij} \end{bmatrix};\ \boldsymbol{\Theta}_{ij} = \begin{bmatrix} \boldsymbol{C}_i \delta_{ij} \boldsymbol{B}_{ij} & \cdots & \boldsymbol{0} \\ \boldsymbol{C}_i \delta_{ij}\left(\boldsymbol{A}_i \boldsymbol{B}_{ij} + \boldsymbol{B}_{ij}\right) & \cdots & \boldsymbol{0} \\ \vdots & \ddots & \vdots \\ \sum_{k=0}^{Hp-1} \boldsymbol{C}_i \boldsymbol{A}_i^{k} \delta_{ij} \boldsymbol{B}_{ij} & \cdots & \sum_{k=0}^{Hp-Hu} \boldsymbol{C}_i \boldsymbol{A}_i^{k} \delta_{ij} \boldsymbol{B}_{ij} \end{bmatrix} (i \neq j) \tag{3.26}$$

The tracking-error vector $\boldsymbol{E}_i(k|k)$ and the optimal solution of (3.24) $\boldsymbol{\Delta U}_i^{*}(k\middle|k)$ for subsystem i are given by the following equations:

$$\begin{aligned} \boldsymbol{E}_i(k\middle|k) &= \boldsymbol{T}_i(k) - \boldsymbol{\Psi}_i \boldsymbol{x}_i(k) - \boldsymbol{\Gamma}_{ii} \boldsymbol{u}_i(k-1) \\ &\quad - \sum_{j\neq i} \boldsymbol{\Gamma}_{ij} \boldsymbol{u}_j(k-1) - \sum_{j\neq i} \boldsymbol{\Theta}_{ij} \boldsymbol{\Delta U}_j(k\middle|k) - \boldsymbol{\Xi}_i(k\middle|k) \end{aligned} \tag{3.27}$$

$$\Delta \boldsymbol{U}_i^*(k|k) = \boldsymbol{K}_{ii}\left[\boldsymbol{\Omega}_i(k|k) - \sum_{j\neq i}\boldsymbol{\Theta}_{ij}\Delta \boldsymbol{U}_j(k|k)\right] \tag{3.28}$$

with $\boldsymbol{K}_{ii} = \left(\boldsymbol{\Theta}_i^T\boldsymbol{Q}_i\boldsymbol{\Theta}_i + \boldsymbol{\Lambda}_i\right)^{-1}\boldsymbol{\Theta}_i^T\boldsymbol{Q}_i$ and $\boldsymbol{\Omega}_i(k|k)$ contains all the right hand side terms in (3.27) except for the term $\sum_{j\neq i}\boldsymbol{\Theta}_{ij}\Delta \boldsymbol{U}_j(k|k)$.

For numerical convenience, equation (3.28) is generally solved for large systems by iterations. In the current work a closed form solution derived from (3.28) is used as follows:

$$\Delta \boldsymbol{U}(k) = \left(\boldsymbol{I} - \boldsymbol{K}_0\right)^{-1}\boldsymbol{K}_1\boldsymbol{\varepsilon}(k|k) \tag{3.29}$$

where

$$\boldsymbol{K}_0 = \begin{bmatrix} \boldsymbol{0} & -\boldsymbol{K}_{11}\boldsymbol{\Theta}_{12} & \cdots & -\boldsymbol{K}_{11}\boldsymbol{\Theta}_{1N} \\ -\boldsymbol{K}_{22}\boldsymbol{\Theta}_{21} & \boldsymbol{0} & \cdots & -\boldsymbol{K}_{22}\boldsymbol{\Theta}_{2N} \\ \vdots & \vdots & \ddots & \vdots \\ -\boldsymbol{K}_{NN}\boldsymbol{\Theta}_{N1} & -\boldsymbol{K}_{NN}\boldsymbol{\Theta}_{N2} & \cdots & \boldsymbol{0} \end{bmatrix}; \boldsymbol{K}_1 = \begin{bmatrix} \boldsymbol{K}_{11} & & \boldsymbol{0} \\ & \ddots & \\ \boldsymbol{0} & & \boldsymbol{K}_{NN} \end{bmatrix} \tag{3.30}$$

By inspection, $\boldsymbol{K}_1$ is a function of the local subsystems gains whereas $\boldsymbol{K}_0$ depends on the gains describing the interactions among the subsystems.

In equation (3.29), the term $\boldsymbol{\varepsilon}(k|k)$ is similar to $\boldsymbol{E}(k|k)$ in (3.12) with:

$$\boldsymbol{T}(k)=\left[\boldsymbol{T}_1(k),...,\boldsymbol{T}_N(k)\right]^T;\ \ \boldsymbol{\Psi}=block-diag\left[\boldsymbol{\Psi}_1,\cdots,\boldsymbol{\Psi}_N\right];\ \boldsymbol{\Gamma}=\begin{bmatrix}\boldsymbol{\Gamma}_{11} & \cdots & \boldsymbol{\Gamma}_{1N}\\ \vdots & \ddots & \vdots\\ \boldsymbol{\Gamma}_{N1} & \cdots & \boldsymbol{\Gamma}_{NN}\end{bmatrix};$$

$$\boldsymbol{L}_C=\begin{bmatrix}\boldsymbol{1}_{\left(H_p ny_1\right)\times 1} & \boldsymbol{0}_{\left(H_p ny_1\right)\times(N-1)}\\ \ddots & \ddots\\ \boldsymbol{0}_{\left(H_p ny_N\right)\times(N-1)} & \boldsymbol{1}_{\left(H_p ny_N\right)\times 1}\end{bmatrix} \tag{3.31}$$

Then the current control moves are calculated as follows:

$$\boldsymbol{\Delta u}(k)=\boldsymbol{K}_{MPC}\boldsymbol{\varepsilon}\left(k\right) \tag{3.32}$$

with $\boldsymbol{K}_{MPC}=\boldsymbol{L}\left(\boldsymbol{I}-\boldsymbol{K}_0\right)^{-1}\boldsymbol{K}_1$, where $\boldsymbol{L}=block\text{-}diag(\boldsymbol{L}_1,...,\boldsymbol{L}_N)$; $\boldsymbol{L}_i=\left[1,0,\cdots,0\right]_{1\times(nu_i Hu)}$.

It should be pointed out here that the nominal stability of the above algorithm requires that the spectral radius of $\boldsymbol{K}_0$ has to be less than 1, i.e. $\left|\rho\left(\boldsymbol{K}_0\right)\right|<1$ where ρ is the spectral radius (Li *et al.* 2005)

The aforementioned formulation for Nash-based distributed MPC strategy can also be used to analyze the specific case of fully decentralized MPC where all the interactions are ignored, i.e. all the terms corresponding to the interaction between the subsystems are eliminated in (3.18)-(3.19), i.e., $\boldsymbol{K}_0$ in (3.32) is omitted. Finally, closed-loop system representations can be obtained for either a Nash-based MPC, a fully decentralized MPC, or a partially decentralized MPC by substituting the control gain $\boldsymbol{K}_{MPC}$ given in equation (3.32) into the state space models presented in (3.14)-(3.15) for

set-point tracking and (3.16)-(3.17) for disturbance rejection. Based on the resulting closed-loop system equations, a performance index can be calculated as will be shown in the next section.

3.3.3 Robust stability and performance

In this subsection we present the definitions and theorems that are required to formulate the proposed methodology. A sufficient condition for robust stability and a measure for robust performance of closed-loop systems are summarized hereafter. In the closed-loop formulations given in (3.14) and (3.16) for set-point and disturbance inputs; respectively, $\boldsymbol{\eta}$ is defined as the state of the system (either $\boldsymbol{\eta}_r$ or $\boldsymbol{\eta}_d$), $\boldsymbol{e}$ as the output error ($\boldsymbol{e}_r$ or $\boldsymbol{e}_d$), and $\boldsymbol{v}$ as the input, i.e. set-point or disturbance($\boldsymbol{r}$ or $\boldsymbol{w}$).

Definition 1 (*Quadratic Lyapunov Stability QLS*) (Boyd *et al.* 1994; Doyle *et al.* 1991) A sufficient condition for asymptotic stability is the existence of $\boldsymbol{P} > 0$, $\boldsymbol{P} = \boldsymbol{P}^T$ and a positive-definite quadratic Lyapunov function $V(k) = \boldsymbol{\eta}(k)^T \boldsymbol{P} \boldsymbol{\eta}(k)$ such that:

$$V(k+1) - V(k) < 0 \tag{3.33}$$

for all admissible $\boldsymbol{A}_{CL}(k) \in Co\left\{\boldsymbol{A}_{CL_1}, \cdots \boldsymbol{A}_{CL_L}\right\}$ and for all initial conditions.

Definition 2 (*Quadratic Lyapunov H_∞ performance QLP*) (Doyle 1991; Gahinet and Apkarian 1994) for closed-loop systems with zero initial state satisfy the above QLS and

$$\left\| \boldsymbol{e}(k) \right\|_{l_2} < \gamma \left\| \boldsymbol{v}(k) \right\|_{l_2} \tag{3.34}$$

for all l_2-bounded input $\boldsymbol{v}$ if there exists $\boldsymbol{P} > 0$, $\boldsymbol{P} = \boldsymbol{P}^T$ and a positive-definite quadratic Lyapunov function $V(k) = \boldsymbol{\eta}(k)^T \boldsymbol{P} \boldsymbol{\eta}(k)$ such that

$$V(k+1) - V(k) + \boldsymbol{e}^T \boldsymbol{e} - \gamma^2 \boldsymbol{v}^T \boldsymbol{v} < 0 \tag{3.35}$$

for all admissible $\boldsymbol{A}_{CL}(k) \in Co\left\{ \boldsymbol{A}_{CL_1}, \cdots \boldsymbol{A}_{CL_L} \right\}$ and for all initial conditions.

The bound γ can be therefore considered as a robust performance index and interpreted as an upper bound on the variability of the closed-loop system due to the effect of either set-point or disturbance input on the output error according to inequality (3.34).

Definition1 can be posed as an LMI problem according to the following theorem:

Theorem 1 The closed-loop system $\boldsymbol{\eta}(\mathrm{k} + 1) = \boldsymbol{A}_{CL}(\mathrm{k})\boldsymbol{\eta}(\mathrm{k})$, $\boldsymbol{\eta}(0) = \boldsymbol{\eta}_0$, and $\boldsymbol{A}_{CL}(\mathrm{k})$ depends affinely on the vertices $\boldsymbol{A}_{CL_i}$ $\forall i \in \{1, \cdots, L\}$, satisfies QLS if there exists a solution to the following system of LMIs:

$$\boldsymbol{A}_{CL_i}^T \boldsymbol{P} \boldsymbol{A}_{CL} - \boldsymbol{P} < \boldsymbol{0} \;\; \forall i \in \{1, \cdots, L\}$$

$$\boldsymbol{P} > 0, \boldsymbol{P} = \boldsymbol{P}^T \tag{3.36}$$

A bound on γ is obtained by solving an LMI optimization problem as described in the following theorem:

Theorem 2 Consider the time-varying closed-loop system given in (3.14) or (3.16), where $\boldsymbol{A}_{CL}$(k) is described above. A sufficient condition for QLP of this system is the existence of $\boldsymbol{P} > 0$, $\boldsymbol{P} = \boldsymbol{P}^T$ such that

$$\begin{bmatrix} \boldsymbol{A}_{CL_i}^T \boldsymbol{P} \boldsymbol{A}_{CL_i} - \boldsymbol{P} & \boldsymbol{A}_{CL_i}^T \boldsymbol{P} \boldsymbol{B}_{CL} & \boldsymbol{C}_{CL}^T \\ \boldsymbol{B}_{CL}^T \boldsymbol{P} \boldsymbol{A}_{CL_i} & \boldsymbol{B}_{CL}^T \boldsymbol{P} \boldsymbol{B}_{CL} - \gamma^2 \boldsymbol{I} & \boldsymbol{D}_{CL}^T \\ \boldsymbol{C}_{CL} & \boldsymbol{D}_{CL} & -\boldsymbol{I} \end{bmatrix} < \boldsymbol{0} \quad \forall i \in \{1, \cdots, L\} \tag{3.37}$$

then minimizing the bound on (3.34) is equivalent to solving:

$$\gamma_{min}^2 = \min_{\boldsymbol{P}} \gamma^2$$

s.t.

$$\begin{bmatrix} \boldsymbol{A}_{CL_i}^T \boldsymbol{P} \boldsymbol{A}_{CL_i} - \boldsymbol{P} & \boldsymbol{A}_{CL_i}^T \boldsymbol{P} \boldsymbol{B}_{CL} & \boldsymbol{C}_{CL}^T \\ \boldsymbol{B}_{CL}^T \boldsymbol{P} \boldsymbol{A}_{CL_i} & \boldsymbol{B}_{CL}^T \boldsymbol{P} \boldsymbol{B}_{CL} - \gamma^2 \boldsymbol{I} & \boldsymbol{D}_{CL}^T \\ \boldsymbol{C}_{CL} & \boldsymbol{D}_{CL} & -\boldsymbol{I} \end{bmatrix} < \boldsymbol{0} \quad \forall i \in \{1, \cdots, L\} \tag{3.38}$$

The proofs of theorems 1 and 2 can be found in (Gao and Budman 2005) and the references therein.

It should be pointed out that the inequality (3.35) can be readily modified to include a penalty term on manipulated variables move such as done in the objective function of the MPC (equation 3.7). Such a modification will lead to a larger LMI than (3.37) since an additional quadratic term corresponding to the manipulated variable cost

would be considered. Also, the addition will result in a more conservative controller since the controller has already been detuned for robustness through proper selection of the weight $\boldsymbol{\lambda}$ as shown in the simulations in section 3.4. Therefore, a penalty term on manipulated variables was not considered in the infinite horizon cost given by equation (3.34).

3.3.4 Proposed Methodology

The LMI problems given in (3.36) and (3.38) can be solved in MATLAB® using MATLAB® LMI solvers available in the Robust Control Toolbox. Problem (3.36) can be solved as a feasibility problem via the function *feasp* whereas problem (3.38) is solved using the *mincx* function.

A controller can be sought that optimizes performance by minimizing the bound γ in (3.38). This controller will ensure that $\left\|\boldsymbol{v}(k)\right\|_{l_2}$ will have the least effect on $\left\|\boldsymbol{e}(k)\right\|_{l_2}$. To find this controller, the input weights ($\boldsymbol{\lambda}$) are optimized to produce the MPC controller that provides the best closed loop performance as follows:

$$\gamma_{opt}^2 = \min_{\boldsymbol{\lambda}\geq\boldsymbol{0}} \gamma_{min}^2 \tag{3.39}$$

where γ_{min} is the solution of the LMI problem (3.38).

All other tuning parameters, such as the prediction and control horizons and the output weights are fixed a priori for simplicity. A smaller value of γ_{opt} implies a better closed loop performance following definition (3.34). Different model structures used by the

distributed MPC strategy are expected to affect the closed-loop performance defined by γ_{opt}. For instance, a decentralized control strategy where the interaction is neglected may result in poor closed-loop performance. Therefore, in order to improve the performance, interaction information has to be considered. On the other hand, decentralized control structures are more favored due to their simplicity, low communication load, and robustness to model uncertainty. In addition to the input weights used as tuning parameters, the binary variables δ_{ij}'s in the decomposed model (3.18)-(3.19) are used to vary the control structure by considering or ignoring interaction terms leading to the entire spectrum of possibilities ranging from a fully decentralized to a fully connected distributed strategy. For N subsystems, the total possible number of theses logic variables n_δ is bounded as $0 \leq n_\delta \leq N(N-1)$. The least possible number of 0 corresponds to a physically decoupled process whereas the maximum possible number of $N(N-1)$ corresponds to a fully interactive system. The user can set some of the binary variables to 0 or 1 a priori and to combine some of them to reduce their number n_δ between the two possible limits. To measure the entire system connectivity that can be also viewed as a measure of control structure simplicity, the following variable is used:

$$\text{Connectivity, } C = \frac{1}{n_\delta} \sum_{i=1}^{N} \sum_{\substack{j=1 \\ j \neq i}}^{N} \delta_{ij} \; ; \quad \delta_{ij} \in \{0,1\} \tag{3.40}$$

Depending on the values of δ_{ij}, the connectivity C ranges from 0 for a fully decentralized structure to 1 for a fully connected distributed structure. Using this connectivity measure,

a trade-off between performance and structure connectivity can be found by solving the following multi-objective MINLP formulation:

$$\min_{\lambda_i,\delta_{ij}} \ \beta\gamma_{opt} + (1-\beta)C$$

$$\boldsymbol{\lambda_i} \geq \boldsymbol{0},\ \delta_{ij} \in \{0,1\},\ \forall\ i,j \in \{1,\ldots,N\} \qquad (3.41)$$

where β is a weighting coefficient $\beta \in [0,1]$ specified by the user to emphasize the contribution of either performance or connectivity. It should be noticed that when communication is not penalized ($\beta = 1$) the optimization does not lead necessarily to fully connected control structure since decentralized control maybe better in the presence of large uncertainty as shown in sections 3.3.2 and 3.3.3. On the other hand when $\beta = 0$ corresponding to the case where the performance is not penalized, the decentralized structure is the preferred one. The optimization problem given above is coded in MATLAB® and solved using a branch and bound approach with modified version of the *fmincon* function from the MATLAB® Optimization Toolbox in which binary variables are considered. Since the algorithm used in *fmincon*, which is a variant of the Sequential Quadratic Programming (SQP) algorithm, obtains only local solutions, a multi-start approach is adopted in which many starting points are generated in an attempt to obtain better solutions close to global optima (Edgar and Himmelblau 2001). The proposed formulation (3.41) searches for the optimal distributed MPC structure to satisfy robust stability and performance constraints that are implicitly considered in the computations of γ_{opt} and at the same time seeks for a simple control structure. The methodology for solving problem (3.41) is summarized as follows:

Step1: Identify the nominal model (3.1)-(3.2) as well as the set of linear models (3.3)-(3.4) that describes the actual process. In this work it is assumed that these models are available. Select and fix the controllers parameters except the input weights λ_i. To decompose the nominal model into N subsystems the number n_δ and the fixed δ_{ij} if there is any, can be chosen as explained above. The coefficient β is also selected based on the importance of each cost in problem (3.41).

Step2: At iteration $p = 0$, initialize the solver with initial guess $\lambda_i^{(0)}, \delta_{ij}^{(0)}$.

Step3: Solve problem (3.41). The closed-loop matrices for either set-point tracking or disturbance rejection problem are updated at each iteration using the formulation presented in (3.14)-(3.17). The stability constraint (3.36) is solved as a feasibility problem. If problem (3.36) is feasible then solve problem (3.38) and obtain γ_{opt} as explained earlier. Obtain the current connectivity C from (3.40) and compute the objective function as weighted sum of γ_{opt} and C with β as weight.

Step4: Stop if convergence criterion is satisfied, otherwise set $p \rightarrow p + 1$; solver selects the next $\lambda_i^{(p)}, \delta_{ij}^{(p)}$ and repeats **Step3**.

As explained earlier, it is expected that finding a global solution of problem (3.41) is not guaranteed thus steps 2-4 may be repeated for different initial guesses to seek for a global solution. The methodology can also be applied to compare different coordination strategies or model decompositions by considering problem (3.39) only for predefined

structures. This is achieved by optimizing each strategy or decomposition outlined in the previous section with respect to the input weights and selecting the one with minimum performance index γ_{opt}. One important issue for future research is the addition of constraints to the problem. To that purpose, since the current analysis requires an explicit solution of the control law, the multi-parametric approach of Bemporad *et al.* (2002) could be used but this could potentially result in computationally expensive problems. In the next section, the methodology is illustrated by its application to three case studies.

3.4 Application of Methodology and Results

3.4.1 Case 1: Model decomposition for decentralized control of a multi-unit process

This first case study involves applying the proposed methodology to solve the *model decomposition problem*. This problem consists in finding the best decomposition of the state space model into subsystems for the design of a fully decentralized control strategy. Thus, for this first case study, centralized control or other type of coordination are not considered. In (Samyudia *et al.* 1994) it was reported that for decentralized control of multiple units there are different model decompositions that can be done while maintaining the same input-output pairings. They defined two methods for decomposition; namely, *physical decomposition* which is based on the physical unit operations and *mathematical decomposition* which is based on the nature of the balance equations and it can be performed across the units. The use of these different decompositions for the design of a decentralized MPC strategy is expected to result in different closed-loop performance. Since both decomposition methods result in a predefined structure, in this case fully decentralized structures, the comparison of their

closed loop performance can be done by solving problem (3.39). A multi-unit process composed of two CSTRs connected in series with a perfect separator is considered (Samyudia *et al.* 1994). The unreacted material is recycled and fed-back to the first reactor. Figure 3.2 shows a simplified flow sheet of the process. The real process is represented by a polytopic model defined by the following two vertices:

$$\begin{bmatrix} dC_1/dt \\ dC_2/dt \\ dT_1/dt \end{bmatrix} = \begin{bmatrix} -1.1002 & 0.4463 & 0 \\ 0.6695 & -1.1369 & 0 \\ 11.7337 & 0 & -0.0214 \end{bmatrix} \begin{bmatrix} C_1 \\ C_2 \\ T_1 \end{bmatrix} + \begin{bmatrix} -0.0368 & 0 \\ 0.0552 & 0 \\ 0 & -0.0026 \end{bmatrix} \begin{bmatrix} F_R \\ P_s \end{bmatrix}$$

$$\begin{bmatrix} C_2 \\ T_1 \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} C_1 \\ C_2 \\ T_1 \end{bmatrix} \tag{3.42}$$

and

$$\begin{bmatrix} dC_1/dt \\ dC_2/dt \\ dT_1/dt \end{bmatrix} = \begin{bmatrix} -1.32 & 0.67 & 0 \\ 0.87 & -1.36 & 0 \\ 12.91 & 0 & -0.0280 \end{bmatrix} \begin{bmatrix} C_1 \\ C_2 \\ T_1 \end{bmatrix} + \begin{bmatrix} -0.066 & 0 \\ 0.094 & 0 \\ 0 & -0.004 \end{bmatrix} \begin{bmatrix} F_R \\ P_s \end{bmatrix}$$

$$\begin{bmatrix} C_2 \\ T_1 \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} C_1 \\ C_2 \\ T_1 \end{bmatrix} \tag{3.43}$$

where C_1 is the concentration in the first reactor, C_2 is the concentration in the second reactor, T_1 is the temperature in the first reactor, F_R is the recycle flow rate, P_s is the steam pressure and the time is in seconds. The objective is to control C_2 and T_1 by manipulating F_R and P_S. To assign equal importance to errors in C_2 and T_1, the errors in T_1 were scaled through division by 100.

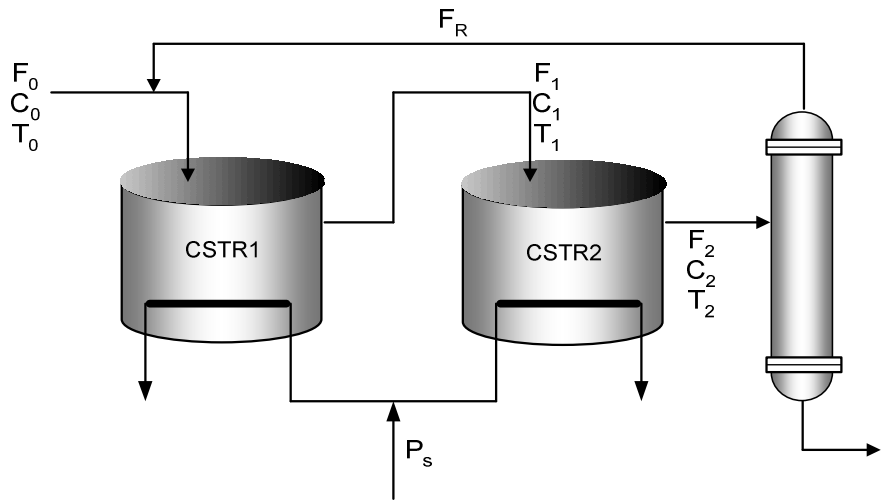


Figure 3.2 Two CSTRs connected in series with a perfect separator (Samyudia *et al.* 1994).

The model given in (3.42) was also used as the nominal model for designing the MPC controllers. The above models were discretized using a sample time of 1 sec. By examining the model, C_2 should be paired with F_R and T_1 should be paired with P_s. In (Samyudia *et al.* 1994) two plant decompositions for decentralized control were proposed; namely, a physical decomposition based on material and energy balances around each reactor, and a mathematical decomposition in which each decomposed subsystem is composed from either the material balances or energy balances of the two units. After ignoring the interactions between the two subsystems the resulting two decompositions are as follows:

1) Physical decomposition:

Subsystem 1:

$$\begin{bmatrix} dC_1/dt \\ dT_1/dt \end{bmatrix} = \begin{bmatrix} -1.1002 & 0 \\ 11.7337 & -0.0214 \end{bmatrix} \begin{bmatrix} C_1 \\ T_1 \end{bmatrix} + \begin{bmatrix} 0 \\ -0.0026 \end{bmatrix} P_s$$

$$T_1 = \begin{bmatrix} 0 & 1 \end{bmatrix} \begin{bmatrix} C_1 \\ T_1 \end{bmatrix}$$

Subsystem 2:

$$dC_2/dt = -1.1369 C_2 + 0.0552 F_R$$

2) Mathematical decomposition:

Subsystem 1:

$$\begin{bmatrix} dC_1/dt \\ dC_2/dt \end{bmatrix} = \begin{bmatrix} -1.1002 & 0.4463 \\ 0.6695 & -1.1369 \end{bmatrix} \begin{bmatrix} C_1 \\ C_2 \end{bmatrix} + \begin{bmatrix} -0.0368 \\ 0.0552 \end{bmatrix} F_R$$

$$C_2 = \begin{bmatrix} 0 & 1 \end{bmatrix} \begin{bmatrix} C_1 \\ C_2 \end{bmatrix}$$

Subsystem 2:

$$dT_1/dt = -0.0214 T_1 - 0.0026 P_S$$

The index γ_{opt} was calculated according to (3.39) for each decomposition to compare their closed-loop performance. Following the calculation of the γ_{opt} numerical simulations were performed with the input weights calculated from problem (3.39) in order to verify the validity of the analysis. To quantify the performance in simulations, an index γ_{sim} was calculated following definition (3.34) as the ratio of the sum of square errors and the sum of squared input changes. The computed values of γ_{opt} and γ_{sim} are summarized in Table 3.1 for step-changes in C_2. The following parameters were used Hp=300, Hu=100, α = 0.6 and $\boldsymbol{Q}_i = \boldsymbol{I}_2$. Equations (3.14)-(3.15) were used to formulate

the closed-loop systems for the set-point tracking problem. Since the analysis produces the worst case scenario, a worst signal for set-point tracking was sought for the purpose of simulation and comparison to the analysis. There is no systematic method to find the signal that will result in the worst γ_{sim} and therefore the search was done by trial and error. Two successive square pulses in set-point of magnitude 1 and -1 respectively and 5 minutes duration were simulated since these signals were found to give large values of γ_{sim}. To approach the actual process behavior, the model representing the actual process was varied in time within the uncertainty values assumed in the analysis. Calculations were done with ($\gamma_{opt,\ uncer}$) and without uncertainty ($\gamma_{opt,\ nom}$). Figures 3.3-3.4 show the controlled output response and Figures 3.5-3.6 show the manipulated variables for the case with uncertainty. Although the analysis is conservative, the simulation results are consistent with the analysis, i.e. the physical decomposition is consistently better than the mathematical decomposition. The conservatism of the analytical results versus the simulation results is due to the fact that the analysis predicts a bound on the worst error whereas the simulations may not necessarily correspond to the worst case scenario which cannot be found in a systematic fashion. While the analysis concludes that both decompositions may have a comparable closed-loop performance when there is no model error the difference in performance becomes clear when uncertainty is introduced. By comparing ($\gamma_{opt,\ nom}$) and ($\gamma_{opt,\ uncer}$) for the two decompositions, when there is a plant-model mismatch it is clear that uncertainty has more effect on the performance of mathematical decomposition. The same conclusion is obtained in simulations by examining γ_{sim} for both decompositions. Figure 3.3 shows that the mathematical decomposition results in a more sluggish concentration response compared to the

physical decomposition. In addition, as seen in Figure 3.4, the control based on mathematical decomposition results in large overshoots in the temperature whereas the control based on physical decomposition results in smaller overshoots.

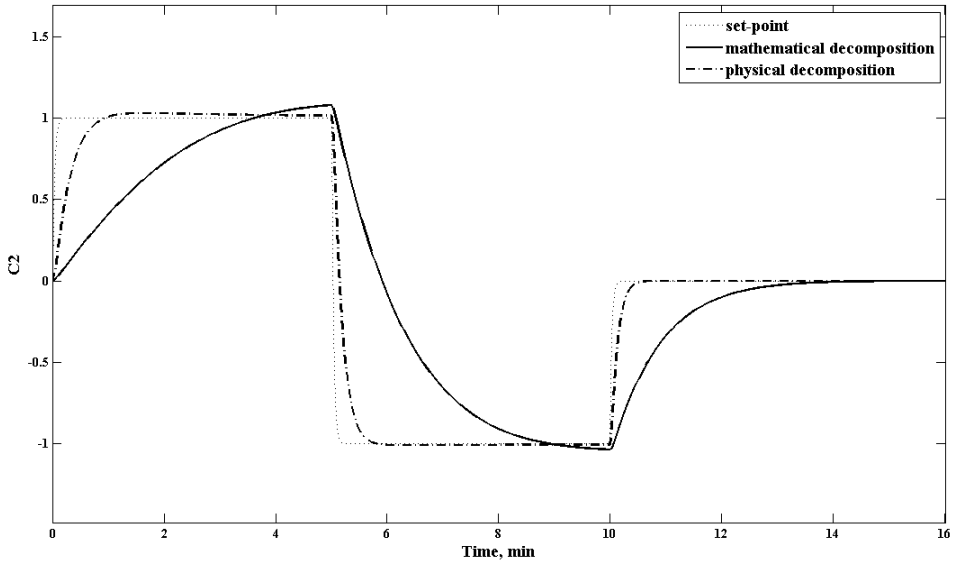


Figure 3.3 Dynamic response of C_2: set-point (dotted line), mathematical decomposition (solid line), physical decomposition (dash-dotted line).

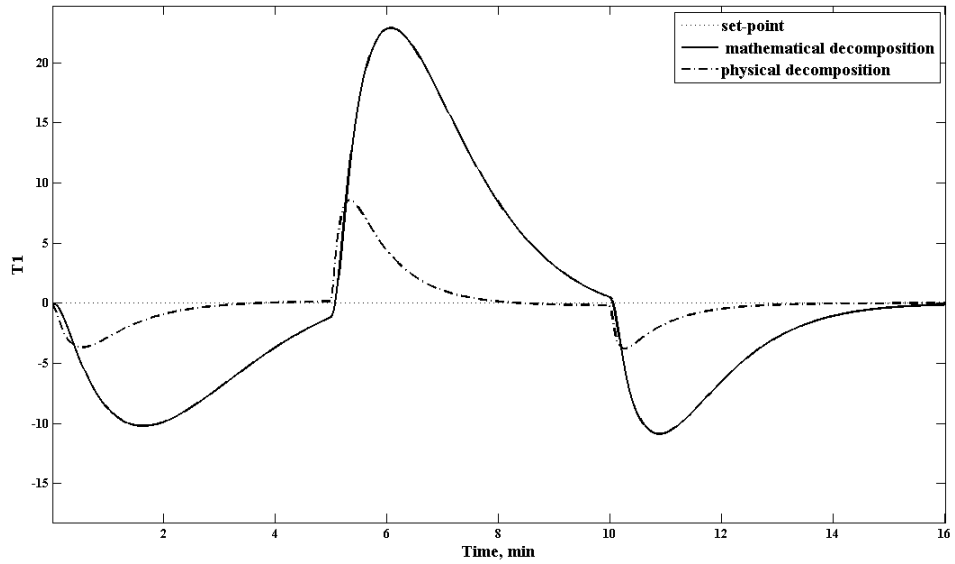


Figure 3.4 Dynamic response of T_1: set-point (dotted line), mathematical decomposition (solid line), physical decomposition (dash-dotted line).

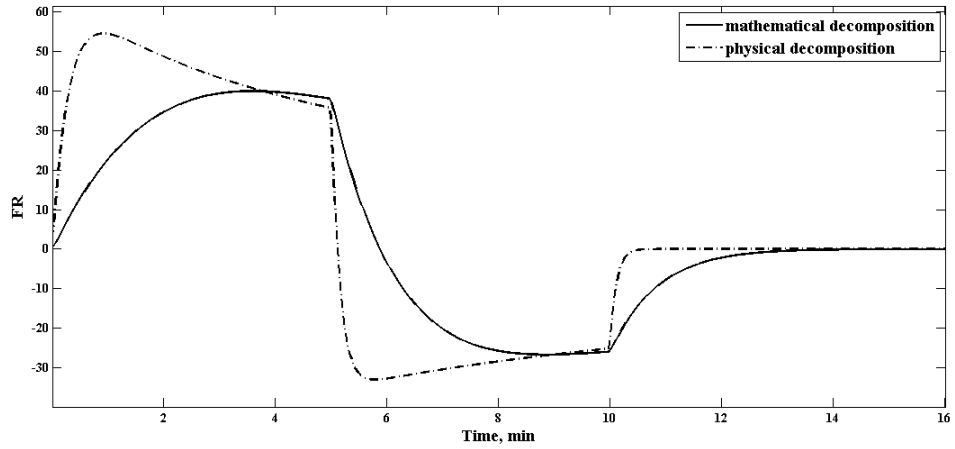


Figure 3.5 Controller output F_R: mathematical decomposition (solid line), physical decomposition (dash-dotted line).

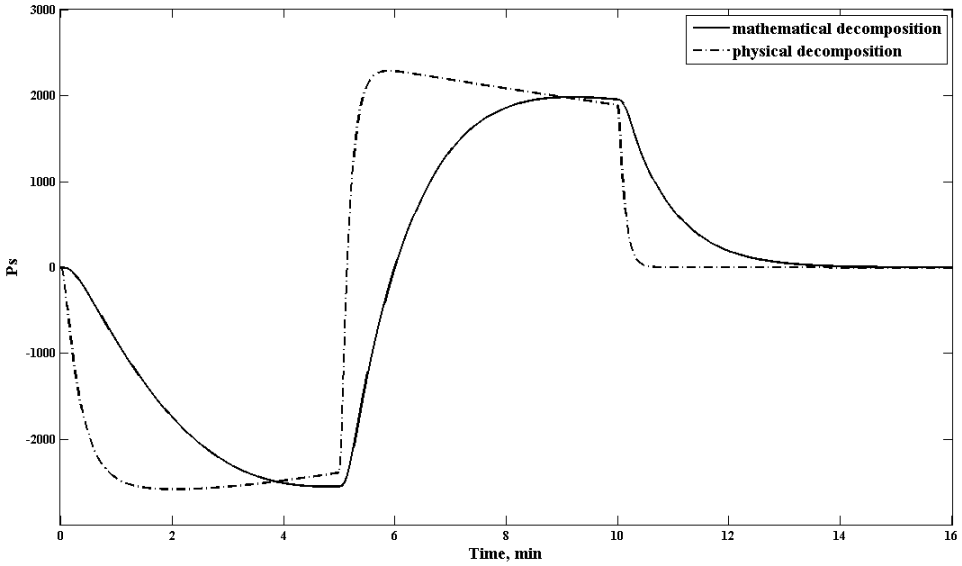


Figure 3.6 Controller output P_s: mathematical decomposition (solid line), physical decomposition (dash-dotted line).

The conclusion that the physical decomposition is better is highly dependent on the magnitude and structure of the uncertainty. For example, it was shown in that for a different uncertainty description the mathematical decomposition is better (Al-Gherwi *et al.* 2008).

Table 3.1 Results of analysis and simulation

	Mathematical decomposition	**Physical Decomposition**
$\gamma_{opt,\ nom}$	0.48	0.47
γ_{sim}	0.21	0.28
λ_{opt}	$\begin{bmatrix} 0.54 & 0 \\ 0 & 3.4\times10^{-2} \end{bmatrix}$	$\begin{bmatrix} 6.6\times10^{-4} & 0 \\ 0 & 1.9\times10^{-2} \end{bmatrix}$
$\gamma_{opt,\ unce}$	1.26	1.07
γ_{sim}	0.62	0.21
λ_{opt}	$\begin{bmatrix} 2.1 & 0 \\ 0 & 4.7\times10^{-2} \end{bmatrix}$	$\begin{bmatrix} 0.25 & 0 \\ 0 & 2.8\times10^{-4} \end{bmatrix}$

3.4.2 Case 2: Comparison of strategies with different degrees of coordination for a high-purity distillation column

To illustrate the use of the proposed methodology for solving the control structure selection problem the example of high-purity column studied in (Skogestad and Morari 1988) is considered. The example is challenging due to the high condition number of the process and its sensitivity to model error. The following nominal state-space continuous-time model is considered:

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{bmatrix} = \begin{bmatrix} -0.0133 & 0 \\ 0 & -0.0133 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} 0.0117 & 0.0115 \\ 0.0144 & 0.0146 \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \end{bmatrix} \qquad (3.44)$$
$$\boldsymbol{y} = \boldsymbol{x}$$

where y_1 and y_2 are the top and bottom product compositions and u_1 and u_2 are the reflux flow-rate and the boil-up. The time is in minutes and the above model is discretized with

a sampling time of 1 minute. To assess the effect of uncertainty, two cases were considered to represent the actual behavior of the process: **case i.** a change of ± 20 % on the steady-state gains in (3.44) such that $u_1=1.2u_{1,model}$; $u_2=0.8u_{2,model}$, **case ii.** a change of ± 80 % on the steady-state gains in (3.44) such that $u_1=1.8u_{1,model}$; $u_2=0.2u_{2,model}$. First a series of predefined control structures was compared by solving problem (3.39). Accordingly, the closed-loop performance, given in terms of γ_{opt}, was calculated for each one of the three MPC strategies: centralized, fully decentralized and Nash-based distributed MPC. The following parameters were assumed for the three controllers as follows: Hp = 20, Hu = 5, α = 0.99 and $\boldsymbol{Q}_i = \boldsymbol{I}_2$. The nominal model in (3.44) can be decomposed into two subsystems using the interactive model (3.18-3.19) as follows:

Subsystem 1:

$$\dot{x}_1 = -0.0133x_1 + 0.0117u_1 + \delta_{12}0.0115u_2$$
$$y_1 = x_1$$

Subsystem2:

$$\dot{x}_2 = -0.0133x_2 + 0.0146u_2 + \delta_{21}0.0144u_1$$
$$y_2 = x_2$$

For fully decentralized MPC $\delta_{12} = \delta_{21} = 0$ whereas for Nash-based distributed MPC $\delta_{12} = \delta_{21} = 1$. By inspecting the two subsystems one can see that the interaction occurs through the inputs only but not through the states. The set-point tracking problem was considered and therefore the closed-loop formulation given in (3.14)-(3.15) was used. The analytical

and simulation results for γ_{opt} and γ_{sim} are summarized in Table 3.2 and the dynamic responses of the system to series of unit set-point changes in y_1 for cases i and ii are shown in figures 3.7-3.10.

Table **3.2** Results of analysis and simulation for different MPC strategies

	Centralized	**Decentralized**	**Nash-Based**
$\gamma_{opt,\ nom}$	0.01	0.33	0.01
γ_{sim}	≈ 0	0.12	≈ 0
λ_{opt}	$\begin{bmatrix} 1.003\times10^{-7} & 0 \\ 0 & 1.05\times10^{-7} \end{bmatrix}$	$\begin{bmatrix} 2.5\times10^{-5} & 0 \\ 0 & 2.5\times10^{-5} \end{bmatrix}$	$\begin{bmatrix} 1.02\times10^{-5} & 0 \\ 0 & 1.02\times10^{-5} \end{bmatrix}$
$\gamma_{opt,\ unce,\ i}$	0.22	0.38	0.19
γ_{sim}	0.10	0.14	0.064
λ_{opt}	$\begin{bmatrix} 3.01\times10^{-6} & 0 \\ 0 & 7.91\times10^{-5} \end{bmatrix}$	$\begin{bmatrix} 4.13\times10^{-4} & 0 \\ 0 & 2.99\times10^{-5} \end{bmatrix}$	$\begin{bmatrix} 8.93\times10^{-3} & 0 \\ 0 & 4.57\times10^{-7} \end{bmatrix}$
$\gamma_{opt,\ unce,\ ii}$	1.99	0.72	2.37
γ_{sim}	0.66	0.36	0.70
λ_{opt}	$\begin{bmatrix} 9.69\times10^{-5} & 0 \\ 0 & 1.57\times10^{-3} \end{bmatrix}$	$\begin{bmatrix} 1.68\times10^{-2} & 0 \\ 0 & 0 \end{bmatrix}$	$\begin{bmatrix} 1.48\times10^{-5} & 0 \\ 0 & 3.54\times10^{-5} \end{bmatrix}$

Similar to the previous example, the results of analysis, even though more conservative, are consistent with the results of the simulation. The results are showing high sensitivity of centralized and Nash-based strategies to model uncertainty ($\gamma_{opt,\ uncer}$). Accordingly, there is a significant increase of errors for the two cases with the uncertainty specified above as compared to the nominal case ($\gamma_{opt,\ nom}$), i.e. the case without uncertainty. In contrast, although the fully decentralized MPC at nominal case

shows poor performance since the interaction information is ignored, the performance did not change by much when the uncertainty is considered (case i). Moreover, for case ii for which a larger uncertainty is considered, the fully decentralized MPC performs better than the other two strategies showing more robustness to model errors. In conclusion, in the presence of model error, especially when this uncertainty is present in the interaction terms, both centralized and Nash-based strategies are more sensitive to model errors than the fully decentralized strategy. Simulation results for case i given in Figures 3.7 and 3.8 show that the performance of both centralized and Nash-based is almost identical and slightly better than fully decentralized performance. Similarly, the simulation results for case ii, shown in Figures 3.9 and 3.10, indicate that the fully decentralized MPC controller outperforms both the centralized and the Nash-based strategies exhibiting smaller overshoots in both y_1 and y_2.

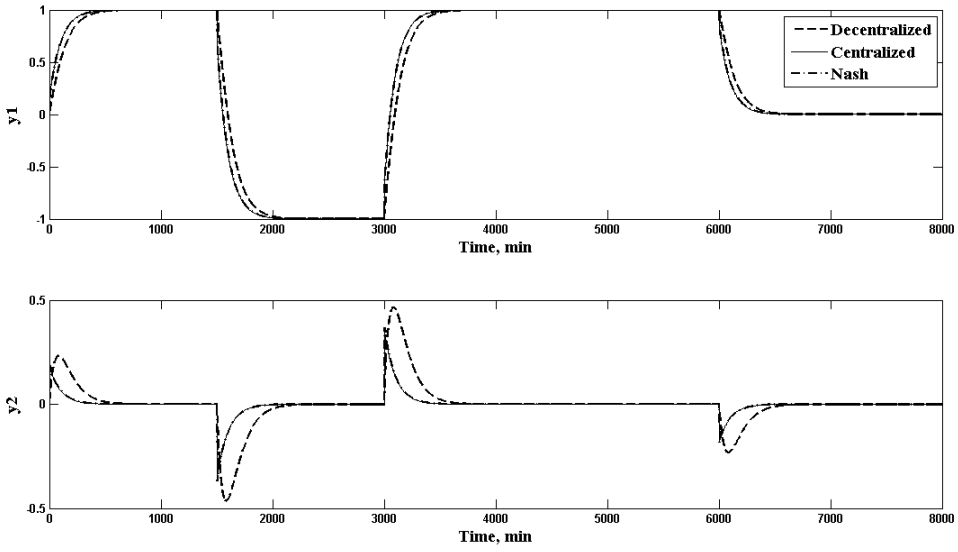


Figure 3.7 Dynamic response of y_1 and y_2 for case i (± 20 % errors): Centralized (solid line), fully decentralized (dashed line), Nash-based (dash-dotted line).

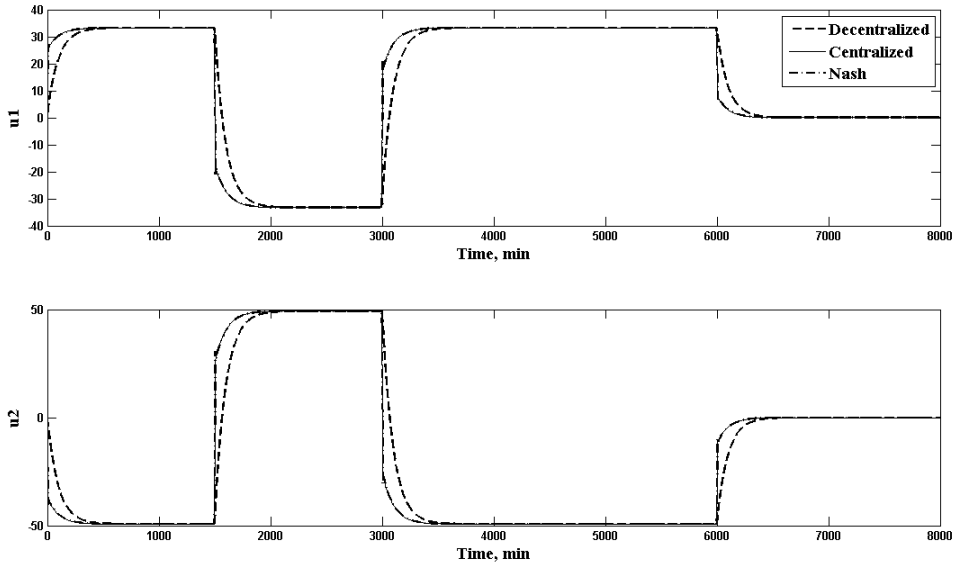


Figure 3.8 Inputs u_1 and u_2 for case i (± 20 % errors): Centralized (solid line), fully decentralized (dashed line), Nash-based (dash-dotted line).

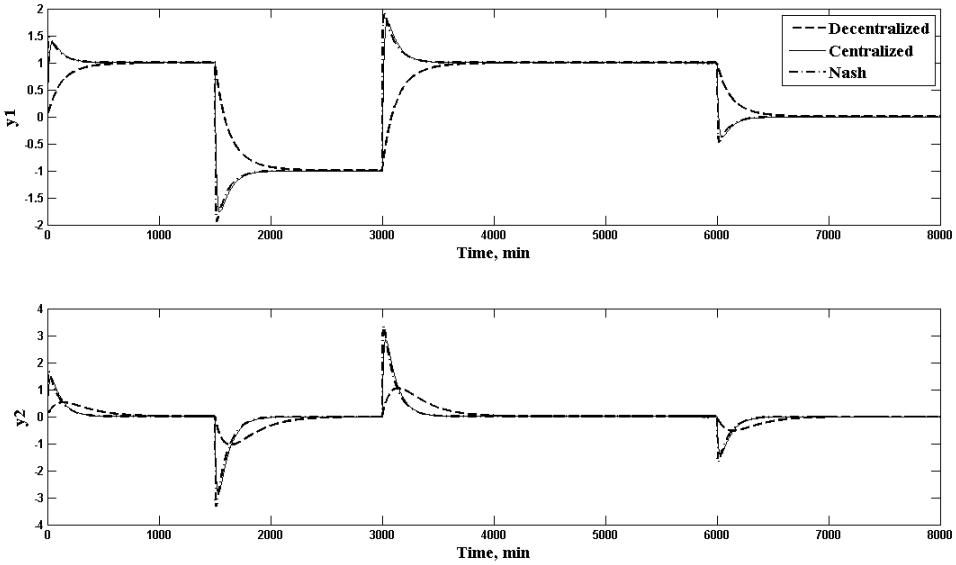


Figure 3.9 Dynamic response of y_1 and y_2 for case ii (± 80 % errors): Centralized (solid line), fully decentralized (dashed line), Nash-based (dash-dotted line).

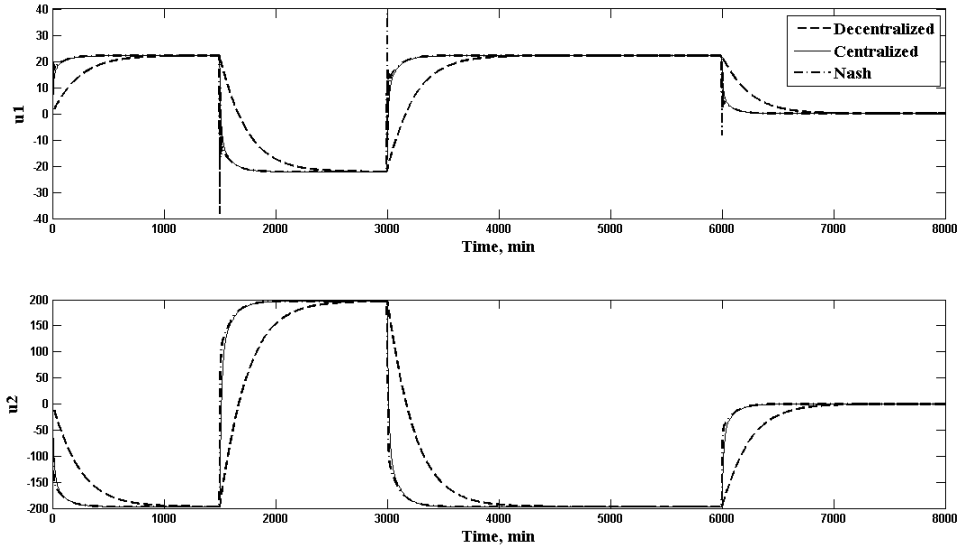


Figure 3.10 Inputs u_1 and u_2 for case ii (± 80 % errors): Centralized (solid line), fully decentralized (dashed line), Nash-based (dash-dotted line).

3.4.3 Case 3: Controller structure selection for a high-purity distillation column

The results obtained from the two previous case studies motivate the need for a methodology that selects the best distributed MPC structure. The methodology outlined in problem (3.41) is used to search for the optimal trade-off between robust closed-loop performance and controller structure complexity. The methodology was applied to the high-purity distillation column of example 2 for the two uncertainty scenarios described in the previous subsection. For the current case study, the variables to be optimized are in addition to the manipulated variables weights $\boldsymbol{\lambda}_{opt}$, the two binary variables δ_{12} and δ_{21} that determined the interactions considered in the controller. The results are summarized in tables 3.3 and 3.4 for the two cases of model errors. Different values of β were used to solve the MINLP in (3.41) in order to see the effect of this weight on the solution. Table 3.3 shows that for the first case the performance index γ_{opt} dominates the solution

resulting in nonzero values for the binary variables up to a weight of $\beta = 0.75$. For $\beta \leq 0.75$, the contribution of the connectivity term C becomes more important leading to the selection of the fully decentralized MPC over the Nash-based or centralized strategies that require full communication. In table 3.4 it is shown that the fully decentralized MPC has already better performance at this level of uncertainty as also found earlier and therefore there is no need to optimize any further by considering the contribution of C.

Table 3.3 Results for case i with ± 20 % errors

β	λ_{opt}	δ_{ij}	γ_{opt}	C	Total cost
1.0	$\begin{bmatrix} 8.93\times10^{-3} & 0 \\ 0 & 4.57\times10^{-7} \end{bmatrix}$	[1 1]	0.19	1	0.19
0.9	$\begin{bmatrix} 8.93\times10^{-3} & 0 \\ 0 & 4.57\times10^{-7} \end{bmatrix}$	[1 1]	0.19	1	0.27
0.75	$\begin{bmatrix} 4.13\times10^{-4} & 0 \\ 0 & 2.99\times10^{-5} \end{bmatrix}$	[0 0]	0.38	0.0	0.28

Table 3.4 Results for case ii with ± 80 % errors

β	λ_{opt}	δ_{ij}	γ_{opt}	C	Total cost
1.0	$\begin{bmatrix} 1.68\times10^{-2} & 0 \\ 0 & 0 \end{bmatrix}$	[0 0]	0.72	0.0	0.72

3.4.4 Case 4: Selection of Control Structure for a reactor/separator system

In this study a reactor/separator process is considered that consists of four main unit operations; reactor with preheater, extractor, flash drum, and distillation column (Lee *et al.* 2000). A schematic of the process is depicted in Figure 3.11. The process has five outputs controlled by five manipulated inputs and two input disturbances. Table 3.5

summarizes these inputs and outputs. A dynamic model of the process with 19 state variables was obtained from first principles. Steady-state information can be found in (Lee *et al.* 2000).

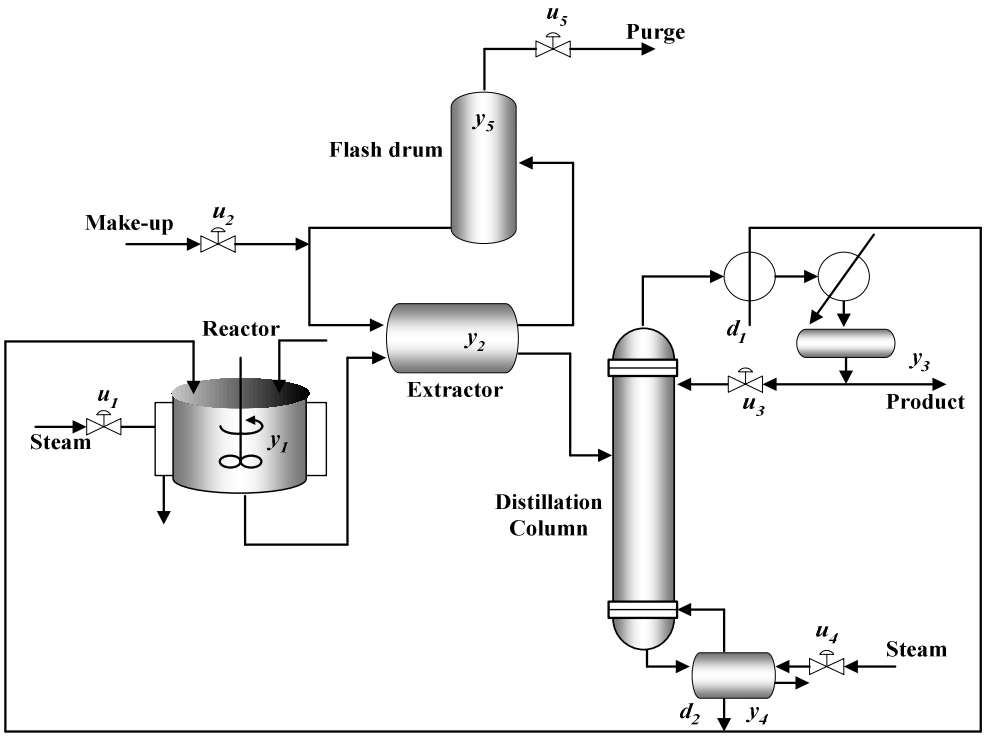


Figure 3.11 Reactor/separator system (Lee *et al.* 2000).

Table 3.5 Inputs and outputs of the reactor/separator process

Controlled outputs	Manipulated inputs	Input disturbances
y_1, reactor temperature	u_1, steam flow rate	d_1, flowrate to the reactor
y_2, raffinate composition	u_2, make-up flowrate	d_2, bottom flowrate
y_3, product composition	u_3, reflux flowrate	
y_4, bottoms composition	u_4, boil-up rate	
y_5, flash-drum pressure	u_5, purge-flowrate	

It is assumed that the two disturbances are unmeasured but they are known to vary between 10 to 20 kmol/hr with a nominal value of 15 kmol/hr. Thus the nominal model, as per the structure defined in (3.1)-(3.2), is obtained by linearizing the nonlinear differential equations around the disturbances nominal flow rates and four other models representing the actual process are also obtained via linearizations around the operating conditions $[d_1,d_2]$ = {[10,10],[10,20],[20,10],[20,20]} which describe the vertices in (3.6). The models are scaled using the steady-state values of the states, inputs, and outputs (Lee *et al.* 2000) The linearized nominal model of the entire process is then decomposed into five subsystems based on the main unit operations. The distillation column is decomposed further into two subsystems one for the rectifying section and one for the stripping section that also includes the feed tray. Table 3.6 shows these subsystems and their local and interaction information. As seen in Table 3.6 the subsystems are not fully interacting and by inspection it can be concluded that there are eight binary variables δ_{ij}. Although subsystem 2 is affected by subsystems 1 and 4, for simplicity the binary variables relating subsystem 2 to subsystems 1 and 4 were merged together reducing the total number of binary variables to 7, i.e. $\{\delta_{14}, \delta_{21}, \delta_{32}, \delta_{34}, \delta_{42}, \delta_{43}, \delta_{52}\}$ where $\delta_{21}=\delta_{24}$. The following parameters were set for the analysis $Hp = 10$, $Hu = 2$, $\alpha_w = 0.99$ and $\boldsymbol{Q}_i = \boldsymbol{I}_5$. The closed-loop systems were formulated for the disturbance rejection problem according to equations (3.16)-(3.17). It should be pointed out that the MATLAB® balanced realization routines *balreal* and *modred* for model reduction were employed to reduce the size of the resulting closed-loop system since the original size is not minimal and this was found to affect the numerical accuracy of the LMI computations. The outputs obtained from the reduced model were found in good match to those obtained

from the full model with average error of (< 1%) between the full model and reduced models.

The methodology explained earlier was applied for β = 0.9 and the results are summarized in Table 3.7. The results show that all the interactions can be ignored except for δ_{34} and δ_{43} that represent the interactions in the distillation column unit. The methodology indicated that ignoring the interaction information in the column system resulted in the violation of the stability constraint (3.36) for any selection of input weights. This conclusion was confirmed by simulation. To simulate the process, d_1 and d_2 are changed as shown in Figure 3.12. The nonlinear differential equations representing the process were simulated using the Euler method with a sampling time of 0.001 hr. The performance of the distributed MPC structure that resulted from the application of the methodology is compared with a centralized MPC in terms of γ_{opt} and found to be 0.58 and 0.55; respectively. Similarly, the performance obtained from simulation γ_{sim} for both distributed MPC and centralized MPC is found to be 0.27 and 0.26; respectively. The analysis indicates that centralized MPC may result in slightly better performance over the distributed MPC with the structure obtained above. The simulation results show, similar to the analysis, a very small difference in performance which justifies the simple structure used in the proposed distributed MPC strategy. The responses of the controlled outputs y_1-y_4 are almost identical for both strategies except y_5 where centralized MPC shows a slightly better performance as shown in Figure 3.13. Thus, it can be concluded that the proposed methodology provides a solution that achieves a trade-off between performance and structure complexity.

Table 3.6 Subsystems information (# refers to the state number in the dynamic model)

Subsystem	Local states x_{ii}	Effect of states from other subsystems x_{ij}	Local inputs u_i	Effect of inputs from other subsystems u_j
1. reactor+preheater	x_{11}=#1-4,19 (y_1 = #3)	x_{14}=#16	u_1	u_4
2.extractor	x_{22}=#5,6 (y_2 = #5)	x_{21} = #1,2 x_{24}=#16	u_2	
3. distillation (rectifying section)	x_{33}=#7-11 (y_3 = #7)	x_{32} = #5 x_{34}=#12,16	u_3	u_2, u_4
4. distillation (stripping section including feed tray)	x_{44}=#12-16 (y_4 = #16)	x_{42} = #5,6 x_{43}=#11	u_4	u_2, u_3
5. flash-drum	x_{55}=#17,18 (y_5 = #18)	x_{52} = #5	u_5	u_2

Table 3.7 Results of a reactor/separator system

β	λ_{opt}	δ_{ij}	γ_{opt}	C	Total cost
0.9	[0.12 2.02 4.10 3.72 0.4] I_5	[0 0 0 1 0 1 0]	0.58	0.29	0.55

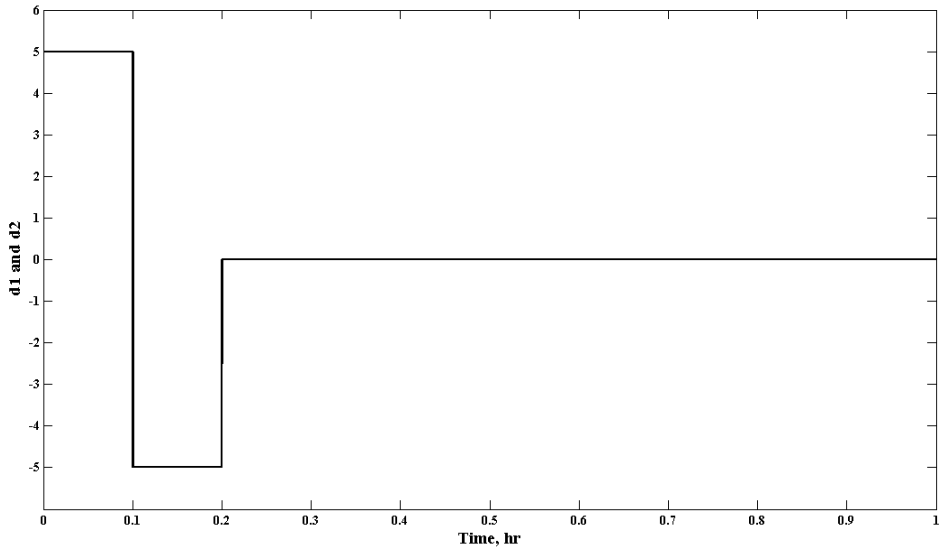


Figure 3.12 Input disturbances used in simulation.

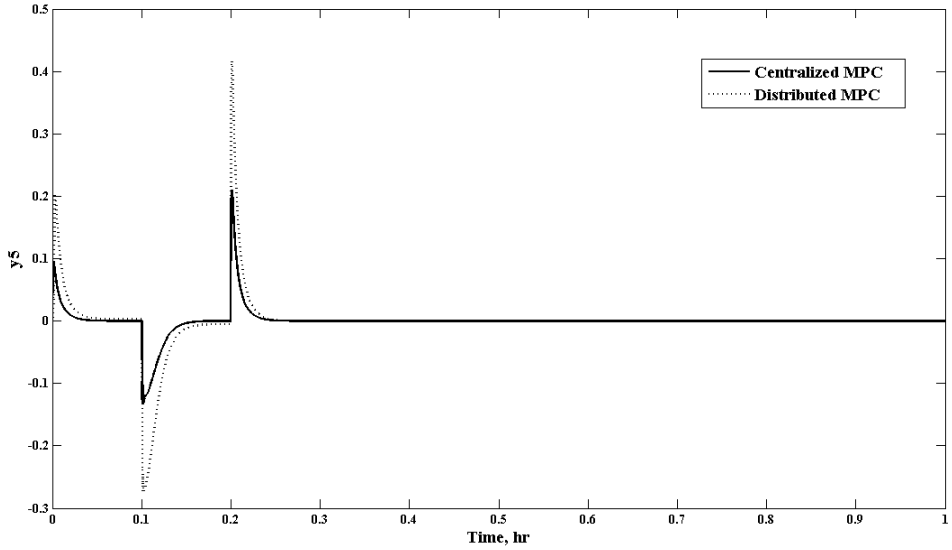


Figure 3.13 Dynamic response of y_5: Centralized (solid line), Distributed MPC (dotted line)

3.5 Conclusion

In this chapter a new methodology has been proposed for comparing distributed MPC strategies with different degrees of coordination in the presence of model error. The technique is based on the quantification of the robust performance based on a variability index that can be calculated by a system of LMI's. This index was also used within an MINLP formulation to search for an optimal tradeoff between robust performance and controller structure complexity. A simple connectivity index was used within the MINLP formulation to represent the cost of interactions considered within the distributed MPC strategy. Four case studies were considered to illustrate the methodology and the simulation results were consistent with the conclusions obtained from the analysis.

CHAPTER 4

A Robust Distributed Model Predictive Control Algorithm

Adapted from Al-Gherwi et al. (2009) and parts of this chapter were submitted to Industrial & Engineering Chemistry Research

4.1 Overview

Distributed Model Predictive Control (DMPC) has received significant attention in the literature. However, the robustness of DMPC with respect to model errors has not been explicitly addressed. In this chapter, a novel online algorithm that deals explicitly with model errors for DMPC is proposed. The algorithm requires decomposing the entire system into N subsystems and solving N convex optimization problems to minimize an upper bound on a robust performance objective by using a time-varying state-feedback controller for each subsystem. Simulations examples are considered to illustrate the application of the proposed method.

4.2 Introduction

Distributed model predictive control (DMPC) has received significant attention in the literature in recent years. The key potential advantages of DMPC are: i) it can provide better performance than fully decentralized control especially when the interactions ignored in the latter approach are strong, and ii) it can maintain flexibility with respect to equipment failure and partial plant shutdowns that may jeopardize the successful operation of centralized MPC. The basic idea of DMPC is to partition the total system of states, controlled and manipulated variables into smaller subsystems and to assign an MPC controller to each subsystem. The design of all the reported DMPC strategies is composed of three parts: (1) **Modeling**: each controller has access to a local dynamic

model of the corresponding subsystem along with an interaction dynamic model that represents the influence of the other subsystems. These models can be obtained by directly decomposing a centralized model of the process (Rawlings *et al.* 2008). (2) **Optimization:** each MPC solves a local optimization problem. Some reported strategies use modified objective functions that take into account the goals of other controllers to achieve full coordination (Venkat 2006; Zhang and Li 2007) whereas some others use strictly local objectives (Li *et al.* 2005), e.g. a Nash-equilibrium objective. (3) **Communication**: at every control time interval all the controllers exchange their respective solutions. These three steps are executed at each time interval in an iterative manner until convergence among the controllers is reached. Venkat (2006) showed that increasing the iterations allows the DMPC strategy to reach the optimal centralized solution while the termination at any intermediate iteration maintains system-wide feasibility. Zhang and Li (2007) analyzed the optimality of the iterative DMPC scheme and derived the closed-form solution for an unconstrained DMPC and showed that it is identical to the centralized MPC solution. Several related strategies appear in the literature. Motee and Sayyar-Rodsari (2003) proposed an algorithm for optimal partitioning of the process model into subsystems to be used with distributed MPC. In that work an unconstrained distributed MPC framework was used and then a weighting matrix was defined to convert the distributed system into a directed graph. Al-Gherwi *et al.* (2010) proposed a methodology for selecting the control structure in the context of distributed model predictive control that achieves a trade-off between closed-loop performance in the presence of model uncertainty and structure simplicity by solving a mixed integer nonlinear program (MINLP). Aiming at reducing the computationally

demanding quadratic dynamic matrix control (QDMC), a decentralized QDMC algorithm was proposed by Charos and Arkun (1993). In this algorithm, it was assumed that the effect of other subsystems on a particular local controller is kept unchanged from the previous sampling time so iterations were not required leading to a significant reduction in computations but with loss in performance. Katebi and Johnson (1997) proposed a decomposition-coordination scheme for generalized predictive control. Jia and Krogh (2001) explored a distributed MPC strategy in which the controllers exchange their predictions and incorporate this information in their local policies. Camponogara *et al.* (2002) discussed the distributed MPC problem and reported an algorithm for cooperative iteration. In addition, the authors proposed heuristics for handling asynchronous communication problems and studied the stability characteristics of distributed MPC. Mercangöz and Doyle (2007) proposed a distributed model predictive estimation and control framework. Liu *et al.* (2009) proposed a distributed MPC scheme for nonlinear systems by designing two Lyapunov-based MPC controllers one to guarantee stability and the other one to enhance the performance. The proposed scheme requires the controllers to communicate only once in a sequential manner at each sampling interval. Based on the Dantzig-Wolfe decomposition and a price-driven approach, a distributed MPC framework for steady-state target calculations was proposed by Cheng *et al.* (2007 & 2008). A comprehensive review on distributed MPC has been recently presented by Scattolini (2009).

The common feature of the reported strategies is that they employ a nominal model of the plant and rely on feedback to compensate for plant-model mismatch.

However, plant–model mismatch may have a significant impact on stability and performance. Thus, the robustness of DMPC strategies to model errors has been identified as a key factor for the successful application of DMPC (Rawlings *et al.* 2008). To the authors' knowledge, the robust DMPC literature is very limited although there is a significant work focused on robust MPC in centralized architecture. Kothare *et al.* (1996) proposed a methodology for robust centralized constrained MPC design that maintains robust stability and minimizes a bound on performance in the presence of model errors. The problem is formulated as a convex optimization problem with linear matrix inequalities (LMI) that is solved efficiently using available algorithms (Boyd *et al.* 1994) and can be used for on-line implementations. This method has been recognized as a potential candidate for use in process industry to handle the issue of plant-model mismatch (Qin and Badgwell 2003).

The aim of this chapter is to present an iterative online algorithm for Robust DMPC to be referred thereafter as RDMPC that explicitly deals with model errors. An LMI-based predictive control formulation (Kothare *et al.* 1996) has been modified into an on-line iterative algorithm for RDMPC. It will be shown that the proposed iterative algorithm can be formulated so as to provide stability and to achieve different control objectives. The control objectives considered in the current work correspond to the objectives used for cooperative control, decentralized control and Nash equilibrium. At convergence the cost function value of cooperative control is equal to the cost function of centralized control. When an overall objective is used the iterative algorithm is shown to converge to the centralized controller following iterations. The chapter will also discuss

the computational effort involved in the proposed iterative RDMPC scheme that can be computationally demanding when a large number of iterations have to be conducted until convergence. However, it will be shown that the use of a relatively smaller number of iterations may still offer performance improvement while resulting in acceptable computational effort. This chapter is organized as follows. In section 4.3 the proposed algorithm is presented. The convergence property and feasibility and robust stability are also discussed in this section. The application of the algorithm is illustrated using three case studies in section 4.4. Conclusions are presented in Section 4.5.

4.3 Definitions and Methodology

4.3.1 Models

In this work, it is assumed that the process model is given by a linear time-varying (LTV) model of the form:

$$\boldsymbol{x}(k+1) = \boldsymbol{A}(k)\boldsymbol{x}(k) + \boldsymbol{B}(k)\boldsymbol{u}(k) \tag{4.1}$$

where $\boldsymbol{x} \in \Re^n$; $\boldsymbol{u} \in \Re^m$ are the process states and inputs; respectively. The actual plant can be represented by a family of models which can be mathematically described by a polytopic model as follows:

$$[\boldsymbol{A}(k)\ \boldsymbol{B}(k)] = \sum_{l=1}^{L} \beta_l [\boldsymbol{A}^{(l)}\ \boldsymbol{B}^{(l)}] \quad ; \sum_{l=1}^{L} \beta_l = 1;\ \beta_l \geq 0 \tag{4.2}$$

Each vertex l corresponds to a linear model obtained from linearizing a nonlinear model or identification of a linear model in the neighborhood of a particular operating point. It is assumed that the states are available either through direct measurement or through estimation to all subsystems. The states and manipulated variables in model (4.1) can be decomposed into N subsystems as follows:

$$\begin{bmatrix} \boldsymbol{x}_{11}(k+1) \\ \vdots \\ \boldsymbol{x}_{ii}(k+1) \\ \vdots \\ \boldsymbol{x}_{NN}(k+1) \end{bmatrix} = \begin{bmatrix} \boldsymbol{A}_{11}(k) & \cdots & \cdots & \cdots & \boldsymbol{A}_{1N}(k) \\ \vdots & \ddots & & & \vdots \\ \boldsymbol{A}_{i1}(k) & \cdots & \boldsymbol{A}_{ii}(k) & \cdots & \boldsymbol{A}_{iN}(k) \\ \vdots & & & \ddots & \vdots \\ \boldsymbol{A}_{N1}(k) & \cdots & \cdots & \cdots & \boldsymbol{A}_{NN}(k) \end{bmatrix} \begin{bmatrix} \boldsymbol{x}_{11}(k) \\ \vdots \\ \boldsymbol{x}_{ii}(k) \\ \vdots \\ \boldsymbol{x}_{NN}(k) \end{bmatrix} + \begin{bmatrix} \boldsymbol{B}_{11}(k) & \cdots & \cdots & \cdots & \boldsymbol{B}_{1N}(k) \\ \vdots & \ddots & & & \vdots \\ \boldsymbol{B}_{i1}(k) & \cdots & \boldsymbol{B}_{ii}(k) & \cdots & \boldsymbol{B}_{iN}(k) \\ \vdots & & & \ddots & \vdots \\ \boldsymbol{B}_{N1}(k) & \cdots & \cdots & \cdots & \boldsymbol{B}_{NN}(k) \end{bmatrix} \begin{bmatrix} \boldsymbol{u}_1(k) \\ \vdots \\ \boldsymbol{u}_i(k) \\ \vdots \\ \boldsymbol{u}_N(k) \end{bmatrix} \tag{4.3}$$

where $i \in \{1, \cdots, N\}$; $\boldsymbol{x}_{ii} \in \Re^{n_{ii}}$; $\boldsymbol{u}_i \in \Re^{m_i}$. For example, in model (4.3) the i^{th} controller for the i^{th} subsystem is to be designed based on the following model:

$$\boldsymbol{x}_i(k+1) = \boldsymbol{A}_i(k)\boldsymbol{x}_i(k) + \boldsymbol{B}_i(k)\boldsymbol{u}_i(k) + \sum_{\substack{j=1 \\ j \neq i}}^{N} \boldsymbol{B}_j(k)\boldsymbol{u}_j(k) \tag{4.4}$$

and using the concept of a polytopic model given in (4.2) it is assumed that the i^{th} subsystem (4.4) can be represented as follows:

$$[\boldsymbol{A}_i(k)\,\boldsymbol{B}_i(k)..\,\boldsymbol{B}_j(k)..] = \sum_{l=1}^{L} \beta_l [\boldsymbol{A}_i^{(l)}\,\boldsymbol{B}_i^{(l)}..\,\boldsymbol{B}_j^{(l)}..] \quad \forall j \in \{1,...,N\},\, j \neq i \qquad (4.5)$$

where $\boldsymbol{x}_i' = \left[\boldsymbol{x}_{11}', \cdots\cdots, \boldsymbol{x}_{ii}', \cdots\cdots, \boldsymbol{x}_{NN}'\right]'$ is the vector of states of subsystem i containing states $\boldsymbol{x}_{ii}$ that can be measured locally augmented with states $\boldsymbol{x}_{jj}$ that are measured in the other subsystems and are communicated among the subsystems. Therefore the matrix $\boldsymbol{A}_i(k)$ contains all the elements of the matrix $\boldsymbol{A}(k)$. Model (4.4) also includes the effect of local controller $\boldsymbol{u}_i$ and the other controllers $\boldsymbol{u}_j$ with corresponding matrices defined as:

$$\begin{aligned} \boldsymbol{B}_i'(k) &= \left[\boldsymbol{B}_{1i}'(k), \cdots\cdots, \boldsymbol{B}_{Ni}'(k)\right]' \\ \boldsymbol{B}_j'(k) &= \left[\boldsymbol{B}_{1j}'(k), \cdots\cdots, \boldsymbol{B}_{Nj}'(k)\right]' \quad \forall j \in \{1,...,N\},\, j \neq i \end{aligned} \qquad (4.6)$$

The general model in (4.4) can be used to represent special limiting cases such as the decentralized case where all the interactions are ignored, e.g. $\boldsymbol{B}_j' = [\boldsymbol{0}]\ \forall j \in \{1,...,N\},\, j \neq i$.

4.3.2 Robust Performance Objective

A formulation for a centralized problem whereby an upper bound on a robust performance objective is minimized was reported by Kothare *et al.* (1996). In the current work a similar formulation is used but the minimization is simultaneously done for every subsystem i defined by (4.4) for which the following min-max problem is solved:

$$\min_{\boldsymbol{u}_i(k+n|k)} \max_{[\boldsymbol{A}_i(k+n)\ \boldsymbol{B}_i(k+n)\ \boldsymbol{B}_j(k+n)],n\geq 0} J_i(k)$$

$$s.t. \qquad (4.7)$$

$$\left|\boldsymbol{u}_i(k+n|k)\right| \leq \boldsymbol{u}_i^{max},\ n \geq 0$$

A local objective $J_i(k)$ can be defined as follows:

$$J_i(k) = \sum_{n=0}^{\infty} [\boldsymbol{x}_i'(k+n|k)\boldsymbol{S}_i\boldsymbol{x}_i(k+n|k) + \boldsymbol{u}_i'(k+n|k)\boldsymbol{R}_i\boldsymbol{u}_i(k+n|k) + \sum_{\substack{i=1 \\ i\neq j}}^{N} \boldsymbol{u}_j^{\bullet\prime}(k+n|k)\boldsymbol{R}_j\boldsymbol{u}_j^{\bullet}(k+n|k)] \qquad (4.8)$$

where $\boldsymbol{S}_i > 0$, $\boldsymbol{R}_i > 0$, $\boldsymbol{R}_j > 0$. The above objective takes into account the goals of the other controllers, third summation in the RHS, in order to achieve the global objective of the entire system. The superscript "•" indicates that the solution was obtained in a previous iteration and remains fixed in the current iteration as will be explained later. In this work the cooperative control problem objective in (4.8) is modified to solve two additional control objectives: Nash equilibrium and decentralized control. Both of these strategies are based on minimizing local objectives of the subsystems. The difference is that for Nash the interaction information is shared among the subsystems while for decentralized control the interaction information is ignored. Accordingly, for both Nash and decentralized control the weights $\boldsymbol{S}_i$ and $\boldsymbol{R}_i$ in (4.8) should be modified. Accordingly, $\boldsymbol{S}_i$ becomes a diagonal matrix where all the diagonal elements are set to zero except the one corresponding to the states of subsystem i and all $\boldsymbol{R}_j$ in the summation term in (4.8) are set to zero except $\boldsymbol{R}_i$.

In practical situations, a decentralized architecture may become a viable option to address communication failures if it can provide stability. On the other hand, for cases where interacting units belong to different companies that agree to exchange information while pursuing local objectives, a Nash equilibrium based strategy may be the scheme of choice.

Since the objective in (4.8) has an infinite time horizon, the problem of finding infinite $\boldsymbol{u}_i$ is computationally intractable. Instead, a state-feedback law is sought for each subsystem *i* as follows:

$$\boldsymbol{u}_i\left(k+n|k\right)=\boldsymbol{F}_{ii}\boldsymbol{x}_{ii}\left(k+n|k\right)+\sum_{\substack{j=1\\ j\neq i}}^{N}\boldsymbol{F}_{ij}\boldsymbol{x}_{ij}\left(k+n|k\right)=\boldsymbol{F}_i\boldsymbol{x}_i\left(k+n|k\right) \tag{4.9}$$

similarly,

$$\boldsymbol{u}_j^{\bullet}\left(k+n|k\right)=\boldsymbol{F}_{jj}^{\bullet}\boldsymbol{x}_{jj}\left(k+n|k\right)+\sum_{\substack{i=1\\ i\neq j}}^{N}\boldsymbol{F}_{ji}^{\bullet}\boldsymbol{x}_{ji}\left(k+n|k\right)=\boldsymbol{F}_j^{\bullet}\boldsymbol{x}_i\left(k+n|k\right) \tag{4.10}$$

Substituting these state-feedback laws in (4.4) leads to the following closed loop model:

$$\boldsymbol{x}_i\left(k+1\right)=\left(\tilde{\boldsymbol{A}}_i\left(k\right)+\boldsymbol{B}_i\left(k\right)\boldsymbol{F}_i\left(k\right)\right)\boldsymbol{x}_i\left(k\right) \tag{4.11}$$

where $\tilde{A}_i\left(k\right)=A_i\left(k\right)+\sum_{\substack{j=1\\ j\neq i}}^{N}B_j\left(k\right)F_j^{\bullet}\left(k\right)$

It is assumed that there exists a quadratic function $V_i(k) = \boldsymbol{x}_i'(k)\boldsymbol{P}_i\boldsymbol{x}_i(k)$, $\boldsymbol{P}_i > 0$, so that, for any plant in (6), this function satisfies the following stability constraint:

$$V_i(k+n+1|k) - V_i(k+n|k) \le -[\boldsymbol{x}_i'(k+n|k)\boldsymbol{S}_i\boldsymbol{x}_i(k+n|k) + \boldsymbol{u}_i'(k+n|k)\boldsymbol{R}_i\boldsymbol{u}_i(k+n|k) + \sum_{\substack{i=1 \\ i\neq j}}^{N} \boldsymbol{u}_j^{\bullet\prime}(k+n|k)\boldsymbol{R}_j\boldsymbol{u}_j^{\bullet}(k+n|k)]$$

$$n \ge 0 \tag{4.12}$$

Substituting (4.11), the robust stability constraint in (4.12) becomes:

$$V_i(k+n+1|k) - V_i(k+n|k) \le -[\boldsymbol{x}_i'(k+n|k)\tilde{\boldsymbol{S}}_i\boldsymbol{x}_i(k+n|k) + \boldsymbol{u}_i'(k+n|k)\boldsymbol{R}_i\boldsymbol{u}_i(k+n|k)] \tag{4.13}$$

where $\tilde{\boldsymbol{S}}_i = \boldsymbol{S}_i + \sum_{\substack{i=1 \\ i\neq j}}^{N} \boldsymbol{F}_j^{\bullet\prime}(k+n|k)\boldsymbol{R}_j\boldsymbol{F}_j^{\bullet}(k+n|k)$

which, for all $n \ge 0$, is given by:

$$\left[\tilde{\boldsymbol{A}}_i(k+n) + \boldsymbol{B}_i(k+n)\boldsymbol{F}_i\right]'\boldsymbol{P}_i\left[\tilde{\boldsymbol{A}}_i(k+n) + \boldsymbol{B}_i(k+n)\boldsymbol{F}_i\right] - \boldsymbol{P}_i + \boldsymbol{F}_i'\boldsymbol{R}_i\boldsymbol{F}_i + \tilde{\boldsymbol{S}}_i \le 0$$

(4.14)

By defining an upper bound, γ_i i.e.

$$J_i(k) \le \boldsymbol{x}_i'(k)\boldsymbol{P}_i\boldsymbol{x}_i(k) = V_i(k) \le \gamma_i \tag{4.15}$$

and substituting the parameterization $\boldsymbol{F}_i = \boldsymbol{Y}_i' \boldsymbol{Q}_i^{-1}$, $\boldsymbol{Q}_i = \gamma_i \boldsymbol{P}_i^{-1}$, followed by a Schur complement decomposition on (4.14) and (4.15), and using the input constraints given in (4.7) it can be shown that the minimization of $J_i(k)$ can be replaced by the minimization of its upper bound γ_i as in the following linear minimization problem with LMI constraints (Kothare *et al.* 1996):

$$
\begin{aligned}
&\min_{\gamma_i, \boldsymbol{Q}_i, \boldsymbol{Y}_i} \ \gamma_i \\
&s.t. \\
&\begin{bmatrix} 1 & \boldsymbol{x}_i'(k) \\ \boldsymbol{x}_i(k) & \boldsymbol{Q}_i \end{bmatrix} \geq 0 \\
&\begin{bmatrix} \boldsymbol{Q}_i & \boldsymbol{Q}_i \tilde{\boldsymbol{A}}_i'^{(l)} + \boldsymbol{Y}_i' \boldsymbol{B}_i'^{(l)} & \boldsymbol{Q}_i \tilde{\boldsymbol{S}}_i^{1/2} & \boldsymbol{Y}_i' \boldsymbol{R}_i^{1/2} \\ * & \boldsymbol{Q}_i & \boldsymbol{0} & \boldsymbol{0} \\ * & * & \gamma_i \boldsymbol{I} & \boldsymbol{0} \\ * & * & * & \gamma_i \boldsymbol{I} \end{bmatrix} \geq 0 \\
&\qquad\qquad \forall l \in \{1, \ldots, L\} \\
&\begin{bmatrix} \left(\boldsymbol{u}_i^{max}\right)^2 \boldsymbol{I} & \boldsymbol{Y}_i \\ \boldsymbol{Y}_i' & \boldsymbol{Q}_i \end{bmatrix} \geq 0
\end{aligned}
\tag{4.16}
$$

The key difference between the centralized control algorithm proposed by Kothare et al. (1996) and the distributed strategy proposed in this work is that every controller in the set $i \in \{1, \cdots, N\}$ solves a local problem as in (4.16) and then the solutions are exchanged in an iterative scheme as will be explained in more details in the next subsection. It should be remembered that one of the reasons to use distributed MPC strategies is to address real time computation issues when dealing with large-scale processes (Li *et al.* 2005). Although the iterations in the proposed scheme may increase the computational time, the problem defined in (16) is numerically advantageous as compared to solving the same problem for the whole system (centralized control). The

reason is that the state feedback controller for each subsystem i is obviously of smaller dimensions than a state feedback controller of the centralized MPC strategy. For instance, $\boldsymbol{Y}_i$ for subsystem i is of dimension $(n \times m_i)$ instead of $(n \times m)$ for centralized system where m is the total number of manipulated variables of the entire process. As shown in the case studies presented later in the manuscript, even with the presence tof the iterative nature of the proposed algorithm implementation can still be performed in real time for many applications.

4.3.3 Robust DMPC Algorithm

This section presents the main result of the current work where an on-line algorithm for RDMPC is proposed. We consider the case where controllers can freely communicate and exchange information. . The algorithm proceeds according to the Jacobi iteration method used for the solution of systems of algebraic equations. The procedure is summarized in *Algorithm 4.1* below.

Algorithm 4.1 (RDMPC)

Step 0 (initialization)*: at control interval k=0 set* $\boldsymbol{F}_i$*=0.*

Step 1 (updating) *at the beginning of control interval (k) all the controllers exchange their local states measurements and initial estimates* $\boldsymbol{F}_i$*'s via communication, set iteration t = 0 and* $\boldsymbol{F}_i = \boldsymbol{F}_i^{(0)}$.

Step 2 (iterations)

while $t \leq t_{max}$

Solve all N LMI problems (4.16) in parallel to obtain the minimizers $\boldsymbol{Y}_i^{(t)}, \boldsymbol{Q}_i^{(t)}$ *to estimate the feedback solutions* $\boldsymbol{F}_i^{(t)} = \boldsymbol{Y}_i'^{(t)} \boldsymbol{Q}_i^{-1(t)}$. *If problem is infeasible set* $\boldsymbol{F}_i^{(t)} = \boldsymbol{F}_i^{(t-1)}$. *Check the convergence for a specified error tolerance* ε_i *for all the controllers*

if $\left\| \boldsymbol{F}_i^{(t)} - \boldsymbol{F}_i^{(t-1)} \right\| \le \varepsilon_i \quad \forall i \in \{1,...,N\}$

break

end if

Exchange the solutions $\boldsymbol{F}_i$*'s and set* $t = t + 1$

end while

Step3 (implementation) *apply the control actions* $\boldsymbol{u}_i = \boldsymbol{F}_i \boldsymbol{x}_i$ *to the corresponding subsystems, increase the control interval* $k = k + 1$, *return to step1 and repeat the procedure.*

Algorithm 4.1 is implemented in MATLAB® and problem (4.16) is solved via MATLAB® LMI solver using the function *mincx*. It should be pointed out that *Algorithm 4.1* computes state-feedback laws for every subsystem therefore full state measurement is assumed. When this assumption is not valid, the design of a state observer is required as will be discussed in a later section. In Step 1 at control interval *k*, the feedback solutions obtained in the previous interval *k-1* are used as initial estimates $\boldsymbol{F}_i^{(0)}$ to start the iterations. Convergence of *Algorithm 4.1* and its stability properties are discussed in the following subsection.

4.3.4 Convergence and Robust Stability Analysis of RDMPC Algorithm

Lemma 1. (Rawlings *et al.* 2008) When a cooperative control objective is used each one of the *N* convex problems defined in *Algorithm 4.1* will converge to the same solution which is the solution of the centralized problem, i.e. $\gamma_1 = \cdots = \gamma_i = \cdots = \gamma_N = \gamma$ where γ is the performance upper bound of centralized MPC.

Proof. For *N* subsystems:

for Subsystem i $\gamma_i^{(t)} = \gamma_i(\boldsymbol{F}_1^{(t-1)}, \cdots, \boldsymbol{F}_i^{(t)}, \cdots \boldsymbol{F}_N^{(t-1)})$

$$= \min_{\boldsymbol{F}_i^{(t)}} \gamma_i(\boldsymbol{F}_1^{(t-1)}, \cdots, \boldsymbol{F}_i^{(t-1)}, \cdots \boldsymbol{F}_N^{(t-1)}) \; i \in \{1, \cdots, N\}$$

similarly for Subsystem j $\gamma_j^{(t)} = \gamma_j(\boldsymbol{F}_1^{(t-1)}, \cdots \boldsymbol{F}_j^{(t)}, \cdots, \boldsymbol{F}_N^{(t-1)})$

$$= \min_{\boldsymbol{F}_j^{(t)}} \gamma_j(\boldsymbol{F}_1^{(t-1)}, \cdots \boldsymbol{F}_j^{(t-1)}, \cdots, \boldsymbol{F}_N^{(t-1)}) \; \forall j \in \{1, \cdots, N\}, \; j \neq i$$

Then from the convexity of problem (4.16) and for any *i* and *j* pair of subsystems,

$\gamma_i^{(t)} \leq \gamma_j^{(t-1)}$ and $\gamma_j^{(t)} \leq \gamma_i^{(t-1)}$

and the reason being that both sides of these two inequalities are using the same value of $\boldsymbol{F}_j = \boldsymbol{F}_j^{(t-1)}$ and $\boldsymbol{F}_i = \boldsymbol{F}_i^{(t-1)}$; respectively. Thus, the γ_i's continue to decrease from iteration to iteration until both inequalities become equalities. Since the minimizations are convex and each is leading to a global optimum, this occurs only when $\boldsymbol{F}_i^{(t)} = \boldsymbol{F}_i^{(t-1)}$ and $\boldsymbol{F}_j^{(t)} = \boldsymbol{F}_j^{(t-1)}$ and consequently $\gamma_i^{(t)} = \gamma_j^{(t)} = \gamma \quad \forall i, j \in \{1, ..., N\}, i \neq j$, i.e.

the minimization with respect to both $\boldsymbol{F}_i$ and $\boldsymbol{F}_j$ gives the same solution which must be, following convexity of problem (4.16), equal to the global optimum of the centralized control problem that has an identical formulation to (4.16).

To prove the robust stability of the proposed algorithm the following definitions are first given:

Definition 1 (*Invariant Set for Quadratic Stability,* Boyd *et al.* 1994) The set $\varepsilon = \left\{ \boldsymbol{x} \in \Re^n \,\middle|\, \boldsymbol{x}'\boldsymbol{Q}^{-1}\boldsymbol{x} \le 1 \right\}$ is said to be an invariant set for $\boldsymbol{x}(k+1) = \Phi(k)\boldsymbol{x}(k)$ where $\Phi(k) = \left(\boldsymbol{A}(k) + \sum_{i=1}^{N} \boldsymbol{B}_i(k)\boldsymbol{F}_i^{(t)}(k) \right)$ if and only if $\boldsymbol{Q}^{-1}$ satisfies $\boldsymbol{Q}^{-1} - \Phi^{(l)\prime}\boldsymbol{Q}^{-1}\Phi^{(l)} \ge 0,\ l \in \{1,\cdots,L\}$. As a result if $x(k) \in \varepsilon$ then $x(k+1) \in \varepsilon$.

Definition 2 (*Intersection of Invariant sets,* Boyd *et al.* 1994). If the sets $\varepsilon_i = \left\{ \boldsymbol{x} \in \Re^n \,\middle|\, \boldsymbol{x}'\boldsymbol{Q}_i^{-1}\boldsymbol{x} \le 1 \right\},\ \forall i \in \{1,\cdots,N\}$ exist then there is a set $\varepsilon = \bigcap_{i=1}^{N} \varepsilon_i$ defined as $\varepsilon = \left\{ \boldsymbol{x} \in \Re^n \,\middle|\, \boldsymbol{x}'\boldsymbol{Q}^{-1}\boldsymbol{x} \le 1 \right\}$ where $0 < \boldsymbol{Q}^{-1} \le \sum_{i=1}^{N} \tau_i \boldsymbol{Q}_i^{-1}$, $\sum_{i=1}^{N} \tau_i = 1$, $0 \le \tau_i < 1$.

Then, the robust stability of *Algorithm1* is given in Theorem 1.

Theorem 1. *At sampling time k and any iteration t > 0, the state feedback solutions* $\boldsymbol{F}_i^{(t)}(k) = \boldsymbol{Y}_i'^{(t)}(k)\boldsymbol{Q}_i^{-1(t)}(k),\ i \in \{1,\cdots,N\}$, *obtained from Algorithm 1, robustly*

asymptotically stabilize the closed loop system $\boldsymbol{x}(k+1)=\Phi(k)\boldsymbol{x}(k)$ *where* $\boldsymbol{A}(k)$ *and* $\boldsymbol{B}_i(k)$ *belong to the polytopic description defined in (4.5)* .

Proof. At time interval *k* the same set of measured or estimated states $\boldsymbol{x}(k)$ is available to all the controllers and at iteration $t > 0$, if the problem posed in (16) is feasible for all subsystems then the condition of definition 2 is satisfied. Accordingly, inequality (14) is satisfied at the intersection and can be written as:

$$\left[\boldsymbol{A}^{(l)}+\sum_{i=1}^{N}\boldsymbol{B}_i^{(l)}\boldsymbol{F}_i^{(t)}\right]'\boldsymbol{P}\left[\boldsymbol{A}^{(l)}+\sum_{i=1}^{N}\boldsymbol{B}_i^{(l)}\boldsymbol{F}_i^{(t)}\right]-\boldsymbol{P}\le-\left[\boldsymbol{S}_i+\sum_{i=1}^{N}\boldsymbol{F}_i^{(t)\prime}\boldsymbol{R}_i\boldsymbol{F}_i^{(t)}\right],\quad \forall i\in\{1,...,N\},\ \forall l\in\{1,...,L\}$$

where $0<\boldsymbol{P}\le\sum_{i=1}^{N}\tau_i\boldsymbol{P}_i^{-1}$, $\sum_{i=1}^{N}\tau_i=1$, $0\le\tau_i<1$.

since $\boldsymbol{S}_i>0, \boldsymbol{R}_i>0$, and $\boldsymbol{P}=\gamma\boldsymbol{Q}^{-1}$ we have

$\boldsymbol{Q}^{-1}-\Phi^{(l)\prime}\boldsymbol{Q}^{-1}\Phi^{(l)}\ge 0,\ l=1,...,N,$ which satisfies definition 1 and thus $\boldsymbol{x}(k+n)$, $n > 0$ belong to the invariant set $\varepsilon=\left\{\boldsymbol{x}\in\Re^n\,\middle|\,\boldsymbol{x}'\boldsymbol{Q}^{-1}\boldsymbol{x}\le 1\right\}$ and $\boldsymbol{x}(\infty)\to 0$

The solution of problem (4.16) at time *k* and iteration *t* if initially feasible then it is also feasible at all future sampling intervals $(k + n)$, $n > 0$. This is because the only constraint that depends on the states is the first constraint in (4.16), i.e. $\boldsymbol{x}(k+n)'\boldsymbol{Q}^{-1}\boldsymbol{x}(k+n)\le 1, n>0$ and the states are given by

$\boldsymbol{x}(k+n)=\left(\boldsymbol{A}(k)+\sum_{i=1}^{N}\boldsymbol{B}_i(k)\boldsymbol{F}_i^{(t)}\right)\boldsymbol{x}(k+n-1), n>0$. This constraint can be shown feasible by using definition 1 for invariant set that is satisfied at time *k* following the same treatment as in Kothare *et al.* (1996).

Remark 1 It should be pointed out that at convergence $\boldsymbol{Q}^{-1}=\boldsymbol{Q}_1^{-1}=\cdots=\boldsymbol{Q}_N^{-1}=\boldsymbol{Q}_{centralized}^{-1}$. This does not hold for the case of Nash since $\boldsymbol{Q}^{-1}\neq\boldsymbol{Q}_{centralized}^{-1}$ and therefore a loss of performance is expected. The reason is that in the Nash scheme each subsystem satisfies its own objective function. A simple example illustrating this point is presented in example1.

Remark 2 Although theoretical convergence of the Jacobi iteration was proven, it was found that numerical noise may exist due to inaccuracies of the LMI solvers in obtaining the solution of problem (4.16). Consequently, to speed up convergence in the presence of this numerical noise when *Algorithm 4.1* is implemented, the *successive Relaxation* (SR) method is employed (Hageman and Young 1981). The SR method is applied to the solution obtained from (4.16) for each subsystem to estimate a weighted average between the current and previous iterate solutions. The method is given by the following recurrence formula:

$$\boldsymbol{F}_i^{(t+1)}=\alpha\bar{\boldsymbol{F}}_i^{(t+1)}+(1-\alpha)\boldsymbol{F}_i^{(t)} \tag{4.17}$$

where α is a parameter to be specified by the user in order to accelerate convergence. $\overline{\boldsymbol{F}}_i^{(t+1)}$ denotes the solution obtained at the current iteration from (4.16) whereas $\boldsymbol{F}_i^{(t+1)}$ is the estimate to be used in the next iteration. Typically, α can be chosen from values between 0 and 2 and when it is set to 1 the original Jacobi iterative scheme is retrieved. Since there is no systematic way to select a value for α in advance, simulations with different values of α were performed to find a suitable value.

4.3.5 RDMPC Algorithm with Output Feedback

It is very common in industrial practice that the states of the system cannot be fully measured due to lack of sensors or because they have no physical meaning as a result of transformation from transfer-matrix description to state-space models. Therefore a state observer is necessary to estimate the states. In this work, *Algorithm1* can still be used but the states $\boldsymbol{x}(k)$ are substituted by their estimates denoted as $\hat{\boldsymbol{x}}(k)$. The observer is designed based on a nominal model of the system [$\boldsymbol{A}$,$\boldsymbol{B}$,$\boldsymbol{C}$] that corresponds to the state space model parameters at the center of the polytopic description given in (4.2). Then, an observer based on the centralized model is embedded with each subsystem that receives all the output measurements and control actions from the other subsystems at interval *k* in order to perform state estimation. The reason behind using a centralized model is to satisfy the condition in Definition 2 that the controllers receive the same state estimates. This estimation is conducted according to the following observer equation:

$$\hat{\boldsymbol{x}}(k+1) = \left(\boldsymbol{A} + \sum_{i=1}^{N} \boldsymbol{B}_i \boldsymbol{F}_i^{(t)} \right) \hat{\boldsymbol{x}}(k) + \boldsymbol{K}\left(\boldsymbol{y}(k) - \boldsymbol{C}\hat{\boldsymbol{x}}(k) \right) \tag{4.18}$$

where $\boldsymbol{F}_i^{(t)}$ is the solutions received and implemented at sampling instant k (t corresponds to either convergence or an intermediate iteration), $\boldsymbol{C}$ is the measurement matrix. The observer gain $\boldsymbol{K}$ is designed such that ($\boldsymbol{A}$-$\boldsymbol{KC}$) is stable. As reported by Wan and Kothare (2002) this can be done by finding $\boldsymbol{K}$ and $\boldsymbol{P}_0 > 0$ such that $\rho^2 \boldsymbol{P}_0 - (\boldsymbol{A} - \boldsymbol{KC}) \boldsymbol{P}_0 (\boldsymbol{A} - \boldsymbol{KC})' \geq 0$ is satisfied where ρ is the minimum decay rate ($0 < \rho < 1$) that can be used as design parameter.

In this work, the stability of observer (4.18) and the controllers obtained from *Algorithm 1* is checked in the simulation using the following closed-loop system that augments the estimated states and the states of the polytopic model (1) (Wan and Kothare 2002):

$$\boldsymbol{\eta}(k+1) = \boldsymbol{A}_{CL}(k)\boldsymbol{\eta}(k) \tag{4.19}$$

where $\boldsymbol{\eta}(k) = \begin{bmatrix} \boldsymbol{x}(k) \\ \hat{\boldsymbol{x}}(k) \end{bmatrix}, \boldsymbol{A}_{CL}(k) = \begin{bmatrix} \boldsymbol{A}(k) & \sum_{i=1}^{N} \boldsymbol{B}_i(k) \boldsymbol{F}_i^{(t)} \\ \boldsymbol{KC} & \boldsymbol{A} + \sum_{i=1}^{N} \boldsymbol{B}_i \boldsymbol{F}_i^{(t)} - \boldsymbol{KC} \end{bmatrix}$.

Then the stability of (4.19) is checked at the vertices of model (4.1) by solving the following feasibility problem (Boyd *et al.* 1994):

$$\text{Find } \boldsymbol{W} > 0,\ \boldsymbol{W} = \boldsymbol{W}' \text{ such that } \boldsymbol{W} - \boldsymbol{A}_{CL}^{(l)'} \boldsymbol{W} \boldsymbol{A}_{CL}^{(l)} > 0 \quad \forall l \in \{1, \ldots, L\} \tag{4.20}$$

where $\boldsymbol{A}_{CL}^{(l)} = \begin{bmatrix} \boldsymbol{A}^{(l)} & \sum_{i=1}^{N} \boldsymbol{B}_i^{(l)} \boldsymbol{F}_i^{(t)} \\ \boldsymbol{KC} & \boldsymbol{A} + \sum_{i=1}^{N} \boldsymbol{B}_i \boldsymbol{F}_i^{(t)} - \boldsymbol{KC} \end{bmatrix}$.

Problem (4.20) is solved as a feasibility problem using the MATLAB® routine *feasp*. In the next section, the application of the proposed iterative algorithm is illustrated using three simulation examples. Although condition (4.20) was always satisfied in the simulation examples considered in the next section observer redesign is necessary if (4.20) is not satisfied and this can be achieved by changing the parameter ρ.

4.4 Case Studies

4.4.1 Example 1

To illustrate remark 1 for the two different schemes, the following system that has 2 states, 2 inputs, and 2 outputs with two vertices is considered (skogestad and Morari 1988):

$$\boldsymbol{A}^{(1)} = \boldsymbol{A}^{(2)} = \begin{bmatrix} 0.9481 & 0 \\ 0 & 0.9481 \end{bmatrix}; \boldsymbol{B}^{(1)} = \begin{bmatrix} 0.0456 & 0.0449 \\ 0.0562 & 0.0569 \end{bmatrix}; \boldsymbol{B}^{(2)} = \begin{bmatrix} 0.0547 & 0.0090 \\ 0.0674 & 0.0114 \end{bmatrix}$$

For the purpose of distributed MPC implementation the system is decomposed into two subsystems. The result of applying *Algorithm 4.1* using RDMPC and Nash schemes at an instant k with $\boldsymbol{x}(k)$ = [-1,0] are shown in Figure 4.1a and 4.1b respectively. The algorithm in both cases converges after three iterations thus illustrating the convergence property proven in Section (4.3.4). Also, as shown in Figure 4.1(a) the two subsystems in RDMPC scheme cooperate and their invariant sets converge to that of centralized MPC whereas in Figure 4.1(b) the two subsystems, controlled by a Nash–equilibrium based scheme

produce an intersection of their corresponding invariant sets which is a subset of the centralized MPC. Thus, a performance loss is expected for the Nash based strategy since the size of the intersection region is correlated to the worst cost as per equation (4.15).

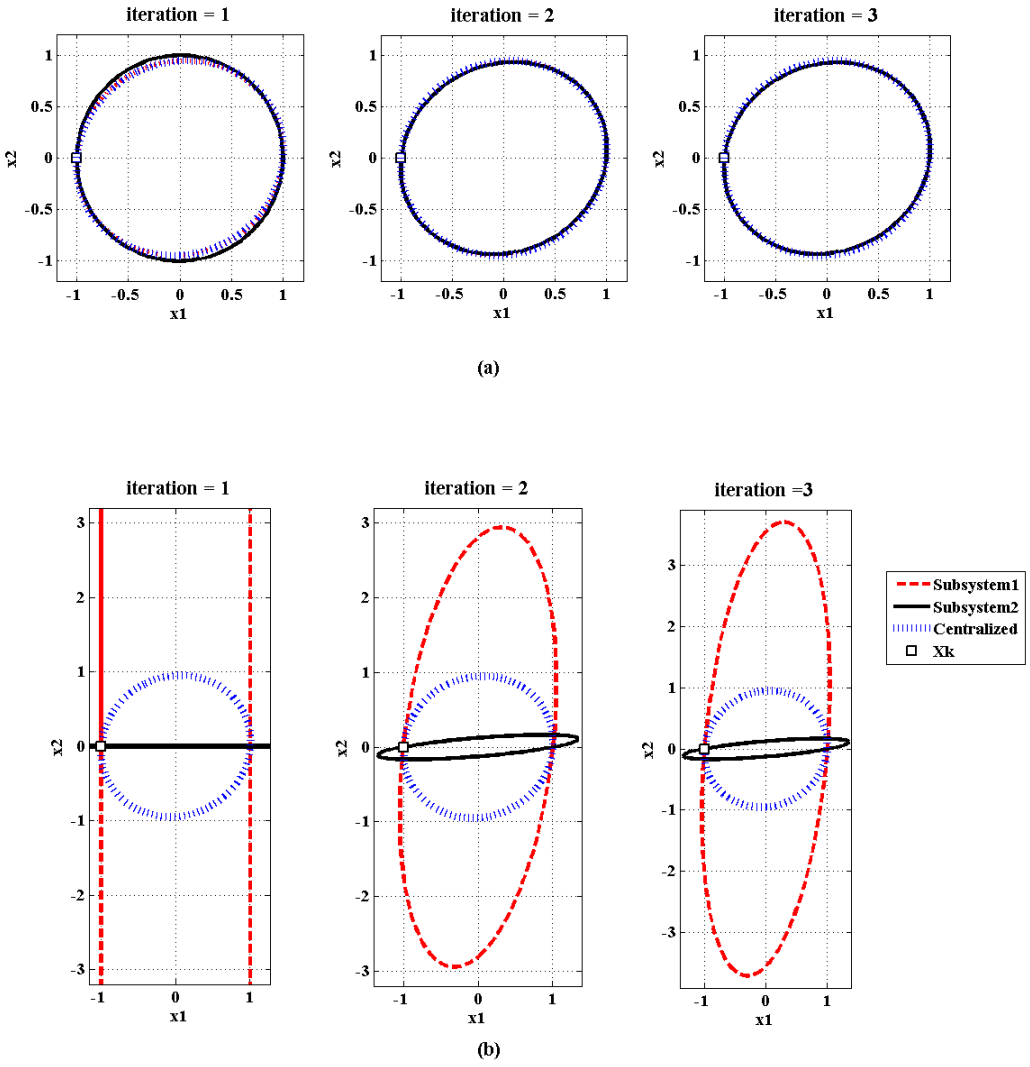


Figure 4.1 Behavior of RDMPC scheme versus Nash; (a) RDMPC (b) Nash

4.4.2 Example 2

A distillation column control problem (Venkat 2006) is considered with uncertainties in the steady-state gains of the model are added to illustrate the robustness of the proposed algorithm. Accordingly, the real process model is assumed to lie within a polytope defined within two vertices defined by the following transfer matrices:

$$\boldsymbol{G}_1(s)=\begin{bmatrix} \dfrac{32.63}{(99.6s+1)(0.35s+1)} & \dfrac{-33.89}{(98.02s+1)(0.42s+1)} \\ \dfrac{34.84}{(110.5s+1)(0.03s+1)} & \dfrac{-18.85}{(75.43s+1)(0.3s+1)} \end{bmatrix};\ \boldsymbol{G}_2(s)=10*\boldsymbol{G}_1(s) \quad (4.21)$$

The corresponding state-space models with 8 states, not shown for brevity, are obtained from a canonical realization of $\boldsymbol{G}_1$ and $\boldsymbol{G}_2$ in (4.21) and included in the appendix. The sampling time used in the simulation is 1 minute. To demonstrate the effectiveness of the proposed method the "bad" pairings, according to the *Relative Gain Array* **RGA** (Bristol 1966), are selected, i.e. the **RGA** element λ_{11} is -1.0874 and accordingly the "bad" pairings are u_1-y_1 (*subsystem1*) and u_2-y_2 (*subsystem2*). The physical constraints on manipulated variables are given by:

$$\left|u_1(k+n)\right| \le 1.5;\ \left|u_2(k+n)\right| \le 2;\ n \ge 0 \quad (4.22)$$

For the purpose of performance comparison between different cases, a cost function is defined as follows:

$$J_{cost} = (1/2Ns)\sum_{k=0}^{Ns}\left(\boldsymbol{x}'(k)\boldsymbol{S}\boldsymbol{x}(k) + \boldsymbol{u}'(k)\boldsymbol{R}\boldsymbol{u}(k)\right) \tag{4.23}$$

where *Ns* is the simulation time, $\boldsymbol{S} = diag(\boldsymbol{S}_i); \boldsymbol{R} = diag(\boldsymbol{R}_i)$. The following parameters are used for the two controllers: $\boldsymbol{S}_{y1} = \boldsymbol{S}_{y2} = 50$ so that $\boldsymbol{S}_i = \boldsymbol{C}_i' \boldsymbol{S}_{yi} \boldsymbol{C}_i + 10^{-6}\boldsymbol{I}$ where $\boldsymbol{C}_i$ is the measurement matrix such that $\boldsymbol{y}_i = \boldsymbol{C}_i\boldsymbol{x}_i$; $\boldsymbol{R}_1 = \boldsymbol{R}_2 = 1$; $\alpha = 0.95$. The value of α is selected, as mentioned above, based on trial and error to speed convergence of the Jacobi iteration. The number of iterations that was required to satisfy the convergence criteria of *Algorithm 4.1* for different values of α is given in Table 4.1. $\alpha = 0.95$ resulted in the fastest convergence.

Table 4.1 Effect of α on convergence with $\varepsilon_1 = \varepsilon_2 = 10^{-3}$

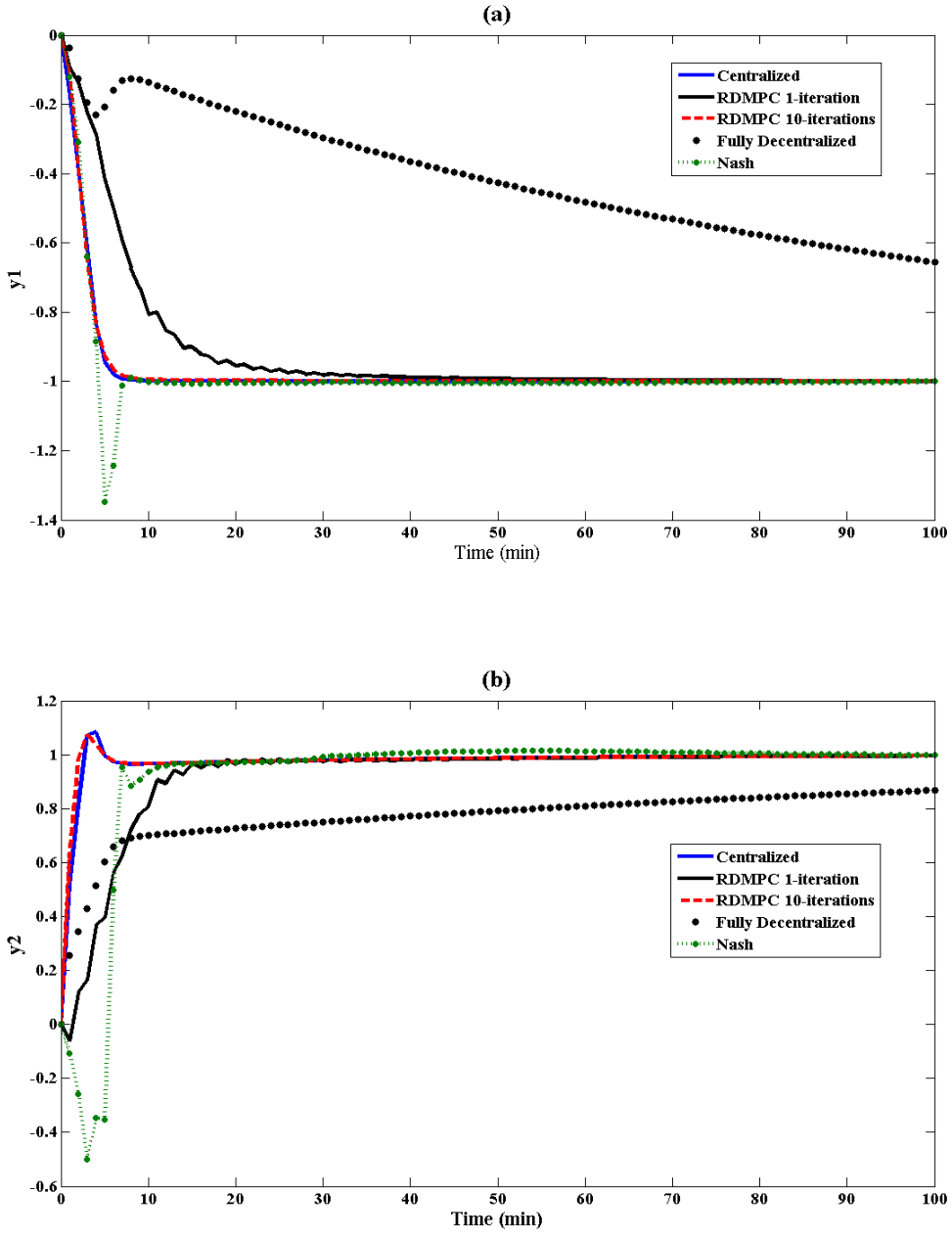
α	**# iterations**
1.05	46
1.00	32
0.95	23
0.90	24
0.70	32

Four cases are considered for the application of *Algorithm 4.1*; fully decentralized, RDMPC with one iteration, RDMPC with 10 iterations, and Nash with 10 iterations. We recall from section 2 that the cost γ decreases monotonically with the number of iterations. Thus, even after one iteration, a performance improvement is expected. The motivation for using a small number of iterations, as mentioned earlier, is

to reduce the amount of computation of the iterative scheme that will be especially critical when dealing with large scale processes. The decentralized strategy used in this example is obtained, as explained in Section 4.3.2, with *Algorithm 4.1* by ignoring interactions in equation (4.4). The performance of *Algorithm 4.1* with these 4 different schemes was compared to the centralized strategy in Figure 4.2. The simulations correspond to simultaneous changes in set-points of both controlled variables y_1 and y_2 by -1 and 1; respectively. A centralized observer was used to perform state estimation as explained in section 4.3.5 and the stability of the augmented closed-loop system (4.19) is verified via the satisfaction of condition (4.20).

In comparison with the centralized scheme, the performance of RDMPC approaches that of the centralized scheme as the number of iterations is increased. The fully decentralized case resulted as expected in the worst performance. A comparison of the cost in (4.23) for the different schemes is given in Table 4.2. This table illustrates that *Algorithm 4.1* can be used, depending on the chosen number of iterations, to obtain a performance that varies between two extremes corresponding to the fully decentralized and the centralized strategies; respectively. It is also clear, from Figures 4.2(c) and 4.2(d), that the constraints given in (4.22) are satisfied. Although the iterative nature of RDMPC results in performance improvement it also leads to an increase in CPU time requirements per control interval. However, since the intermediate iterations were shown to provide acceptable levels of performance then the Algorithm can be potentially terminated at any intermediate iteration before convergence if computation time is an issue. Table 4.3 shows the maximum CPU times per control interval of RDMPC with one and ten iterations. All the simulations were carried out on a Core 2 CPU 1.66 GHz computer. The

CPU time of centralized MPC is in between the two cases. This illustrates the fact that RDMPC can outperform centralized MPC in terms of the CPU time when terminated after a few iterations with small performance loss. It can be also noticed that RDMPC with 10 iterations required 4 seconds of CPU time which is still very reasonable provided that the sampling time is 1 minute so it can still be implemented in real-time.



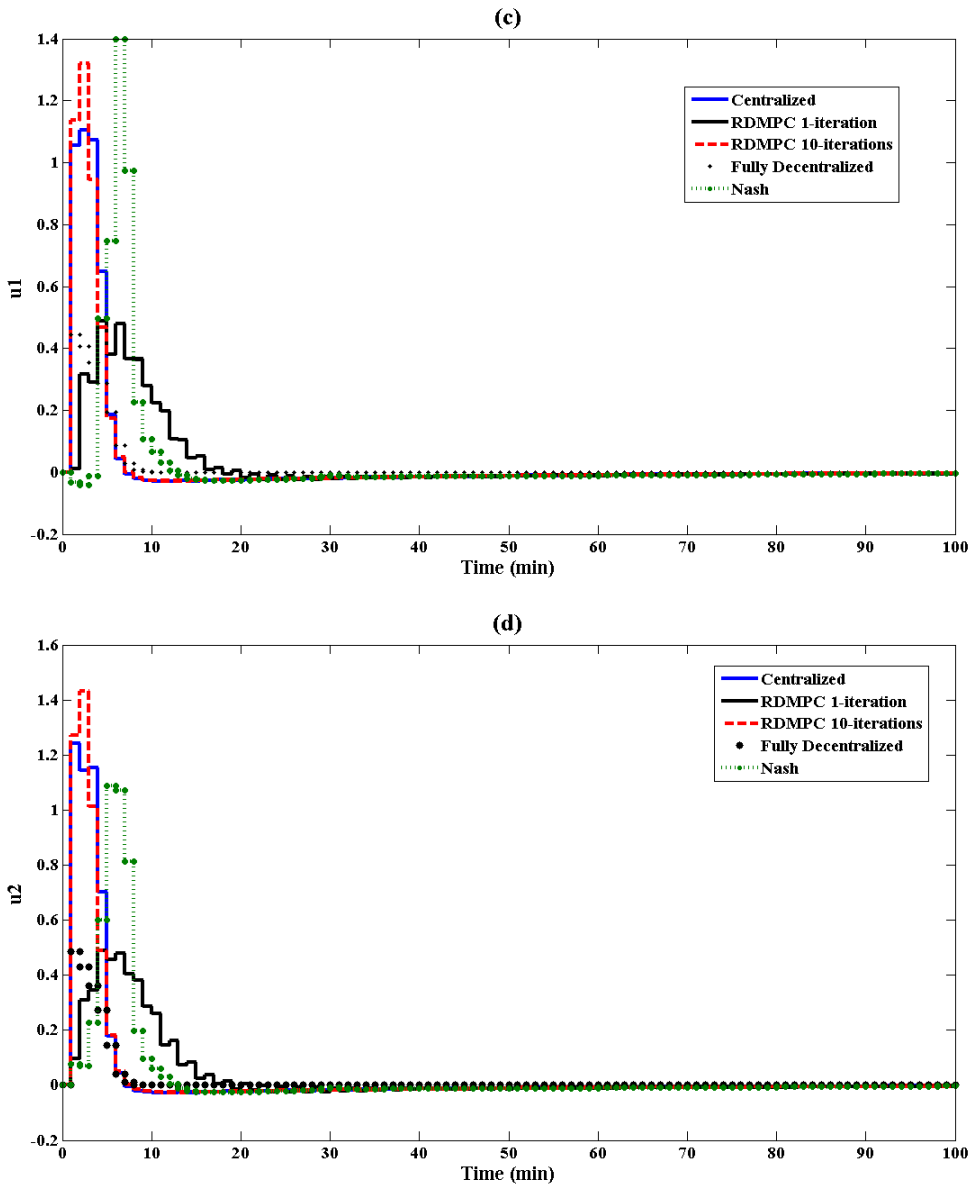


Figure 4.2 Dynamic response of controlled and manipulated variables a) y_1, b) y_2, c) u_1, d) u_2

Table 4.2 Cost for different strategies

Strategy	Cost (4.23)
Centralized	0.92
RDMPC (10 iteration)	0.93
RDMPC (1 iteration)	2.43
Nash (10 iterations)	2.93
Fully decentralized	35.9

Table 4.3 CPU time requirements for RDMPC and Centralized MPC

Iterations	Maximum CPU time, s
1	0.8
10	4.0
Centralized	1.5

4.4.3. Example 3

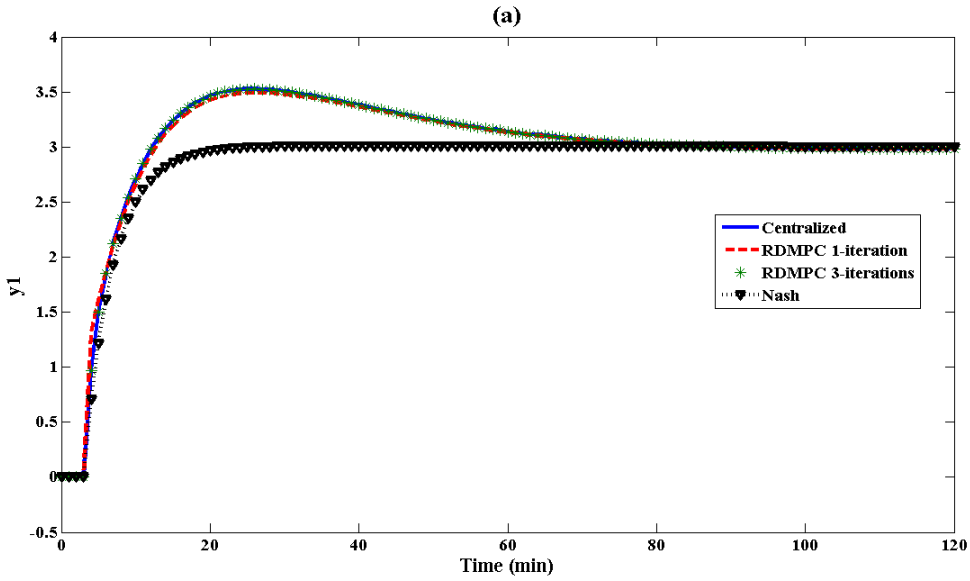
Here a process with 3 inputs and 3 outputs is considered. The real process model is assumed to vary between the following two models:

$$\boldsymbol{G}_1(s)=\begin{bmatrix} \frac{4.05e^{-6s}}{50s+1} & \frac{1.77e^{-6s}}{60s+1} & \frac{5.88e^{-6s}}{50s+1} \\ \frac{5.39e^{-4s}}{50s+1} & \frac{5.72e^{-2s}}{60s+1} & \frac{6.90e^{-2s}}{40s+1} \\ \frac{4.30e^{-4s}}{33s+1} & \frac{4.42e^{-4s}}{44s+1} & \frac{7.20}{19s+1} \end{bmatrix};\quad \boldsymbol{G}_2(s)=0.4\boldsymbol{G}_1(s) \tag{4.24}$$

State-space models of 21 states are obtained based on a canonical realization of equation (4.24) and not shown for brevity. The constraints on manipulated variables are given by $|u_i(k+n)| \leq 10,\ n \geq 0,\ i = 1,2,3$. The system given above was decomposed into three subsystems; *viz.*, y_1-u_1 (*subsystem1*), y_2-u_2 (*subsystem2*), and y_3-u_3 (*subsystem3*). The controllers parameters used in simulation are; $\boldsymbol{S}_1 = \boldsymbol{S}_2 = \boldsymbol{S}_3 = 1$, $R_1 = R_2 = R_3 = 1$, $\alpha = 1$, $\varepsilon_1 = \varepsilon_2 = \varepsilon_3 = 10^{-2}$. A sampling time of 2 minutes was used.

In the application of *Algorithm 4.1*, both RDMPC and Nash schemes were considered. RDMPC was simulated for two different situations: i- the algorithm is terminated after one iteration and ii- the algorithm was left to reach convergence. For the Nash based cost the algorithm was terminated after the convergence criterion was satisfied. Figure 4.3 depicts the performance of *Algorithm 4.1* in terms of output response compared with centralized MPC for the set-point change [$y_1 = 3$; $y_2 = 3$; $y_3 = -3$] and it illustrates that following convergence the RDMPC algorithm results in an identical response to the centralized MPC. Figure 4.4 shows that the constraints on manipulated variables are satisfied. For this example, the RDMPC algorithm converges very quickly in about three iterations after which the error tolerances specified above ($\varepsilon_1 = \varepsilon_2 = 10^{-2}$) are met. The Nash cost based simulations also showed convergence after three iterations but with an apparent loss in performance even when compared to the RDMPC that uses one iteration only. This is, as explained earlier, due to the fact that the invariant set achieved by Nash when it is converged is always smaller than the invariant set corresponding to the Centralized scheme. Figure 4.5 shows the convergent behaviour of the RDMPC algorithm obtained in the first sampling interval. The upper bounds $\gamma_1, \gamma_2,$ and γ_3 for subsystems 1,2, and 3; respectively, obtained by solving (4.16) in parallel and by

applying *Algorithm1*, converge to the same value after about 3 iterations and this value is identical to that obtained for centralized MPC. The cost, defined by equation (4.23), for Centralized, RDMPC (1 iteration), RDMPC (3 iterations), and Nash (3 iterations) were: 0.63, 0.67, 0.63, and 1.60; respectively. Thus RDMPC even when terminated after one iteration it maintains very similar performance to that obtained with Centralized MPC. In addition, the maximum CPU times in seconds for Centralized, RDMPC (1 iteration), RDMPC (3 iterations) were 35 sec, 23 sec, and 60 sec; respectively. Accordingly, RDMPC with one iteration required less CPU time than the centralized scheme whereas the performance was very close to the centralized case. The case of RDMPC with 3 iterations which were found to result in convergence is also reasonable since the sampling time is 2 minutes and therefore the real-time implementation is feasible. It should also be remembered that the simulations were carried out using MATLAB® and a further reduction in CPU time is expected when the algorithm is implemented in a more efficient computing platform.



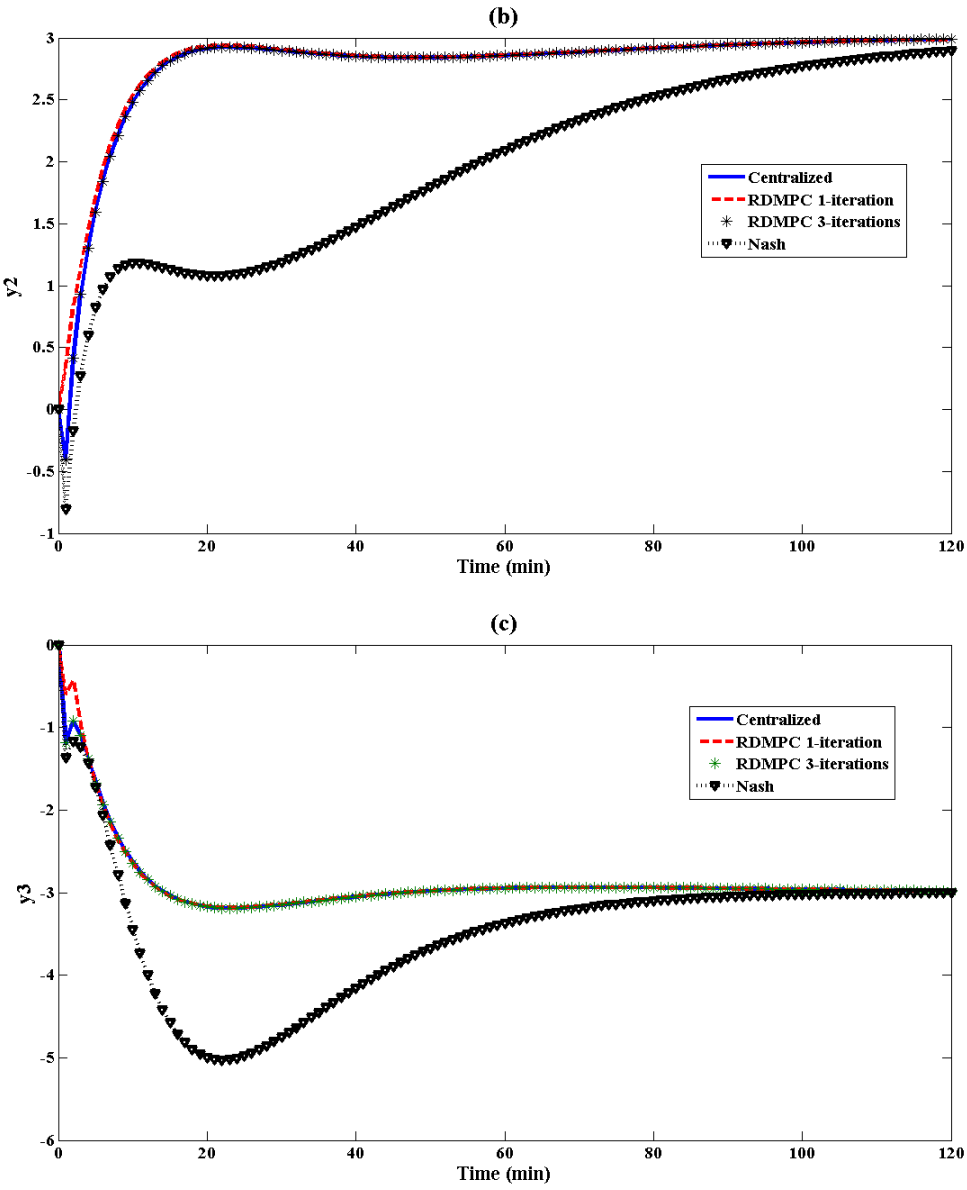
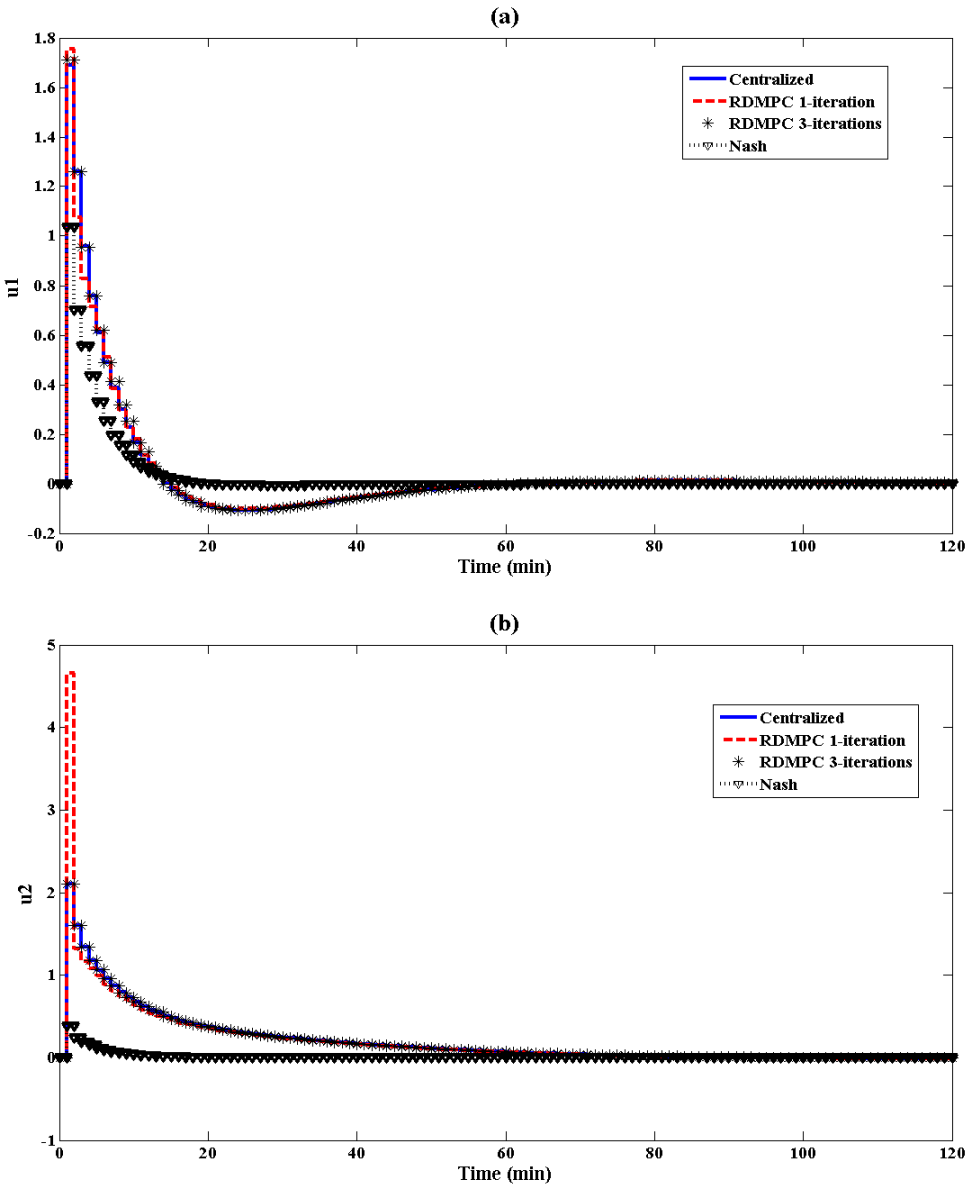


Figure 4.3 Dynamic response of controlled variables a) y_1, b) y_2, c) y_3


(a)
Centralized
RDMPC 1-iteration
RDMPC 3-iterations
Nash
u1
Time (min)
(b)
Centralized
RDMPC 1-iteration
RDMPC 3-iterations
Nash
u2
Time (min)

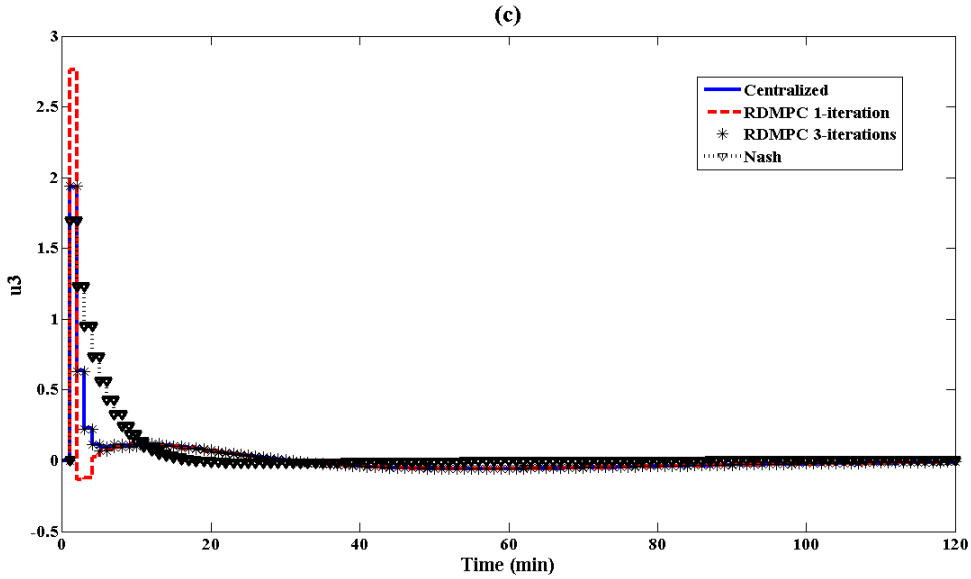


Figure 4.4 Dynamic response of manipulated variables a) u_1, b) u_2, c) u_3

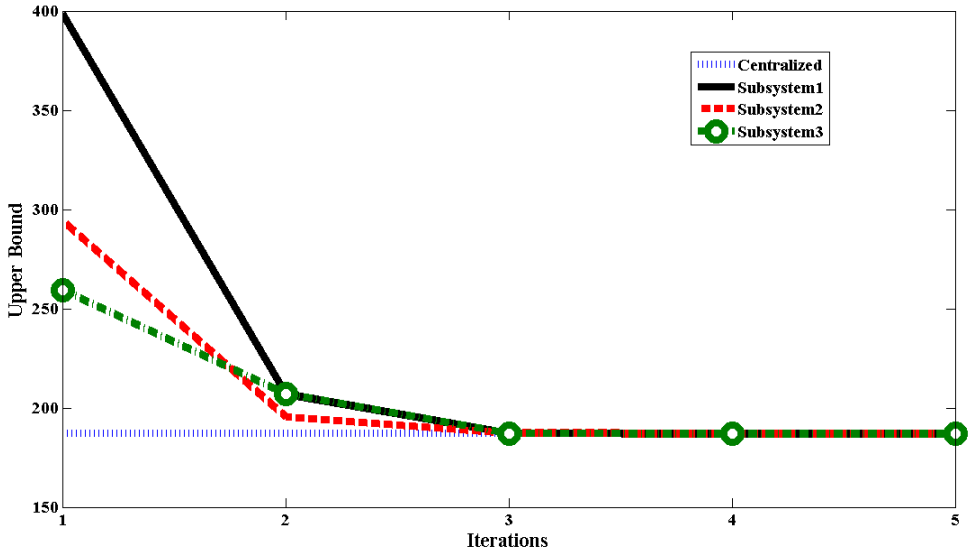


Figure 4.5 Convergence characteristics of Algorithm 4.1 at the first sampling interval.

4.4.4. Example 4

In this simulation example a reactor/separator system (Lee *et al.* 2000) that consists of four unit operations is considered and illustrated in Figure 4.6. The process

has 19 states five of which are controlled variables, five manipulated variables, and two unmeasured disturbances. The summary of the system's inputs and outputs is given in Table 4.4. A nonlinear dynamic model of the process can be found in Lee et al. (2000) as well as steady-state information. In the simulation it is assumed that the disturbances are unknown but both are known to vary in the interval [13,17] where their nominal value is 15 kmol/h. Thus to obtain the vertices in (1)-(2) the model is linearized around the points $[d_1,d_2]$={[13,13],[17,17],[17,13],[13,17]} resulting in 4 linear models which are used to formulate the polytopic model description. Also a nominal model is obtained at the nominal operating point and used in the state estimation as to be explained below. For RDMPC, the system is decomposed into 4 subsystems based on the unit operations. A central observer is used for estimation. In addition to the states, this observer is also used for estimating the disturbance, thus the states' vector in the observer (4.18) is augmented with disturbance states corresponding to the five measured variables that are also controlled variables in this example. Therefore the predicted states are obtained from the following equations:

$$\begin{aligned}\hat{\boldsymbol{x}}(k+1) &= \boldsymbol{A}\hat{\boldsymbol{x}}(k)+\sum_{i=1}^{N}\boldsymbol{B}_i\boldsymbol{u}_i(k)+\boldsymbol{K}_x\left(\boldsymbol{y}(k)-\boldsymbol{C}\hat{\boldsymbol{x}}(k)-\hat{\boldsymbol{d}}(k)\right)\\ \hat{\boldsymbol{d}}(k+1) &= \hat{\boldsymbol{d}}(k)+\boldsymbol{K}_d\left(\boldsymbol{y}(k)-\boldsymbol{C}\hat{\boldsymbol{x}}(k)-\hat{\boldsymbol{d}}(k)\right)\end{aligned} \tag{4.25}$$

where $\hat{\boldsymbol{d}}(k)$ is the estimated disturbance states, $\boldsymbol{K}_x$ and $\boldsymbol{K}_d$ are the observer gains for the state and disturbance; respectively. To remove steady-state offsets, the inputs are calculated form the following equation:

$$\boldsymbol{u}_i(k) = \boldsymbol{F}_i^{(t)}(k)\,\hat{\boldsymbol{x}}(k) + \boldsymbol{u}_i^{ss} \tag{4.26}$$

where $\boldsymbol{u}_i^{ss}$ is computed by solving the nominal steady-state equation at time instant k using the estimated disturbance $\hat{\boldsymbol{d}}(k)$ to ensure offset-free control (Muske and Rawlings 1993). Both state estimation and steady-state computations are performed within the estimator and the results are supplied to the different subsystems. The process is then simulated with the input disturbances shown in Figure 4.7. The sampling time is 0.01 hour and the controllers parameters used in simulation are; $\boldsymbol{S}_1 = 1$ $\boldsymbol{S}_2 = 1000$, $\boldsymbol{S}_3 = 1000\boldsymbol{I}_2$, and $\boldsymbol{S}_4 = 1$, $R_1 = R_2 = R_3 = R_4 = 1$,. Only one iteration is performed for RDMPC algorithm since it provides a very close cost to centralized counterpart. The nonlinear plant was integrated using MATLAB® routine *ode45* and the simulations were performed for Centralized and RDMPC (1 iteration). The responses of the outputs y_1-y_5 are identical in both cases. The costs of centralized and RDMPC (1 iteration) were 0.2727, 0.2729; respectively. Figure 8 shows the response of y_5. The maximum CPU times in seconds for Centralized and RDMPC (1 iteration) were 33.4 and 22; respectively. Thus RDMPC in this case can be advantageous since it provides a very similar performance with lesser computation time.

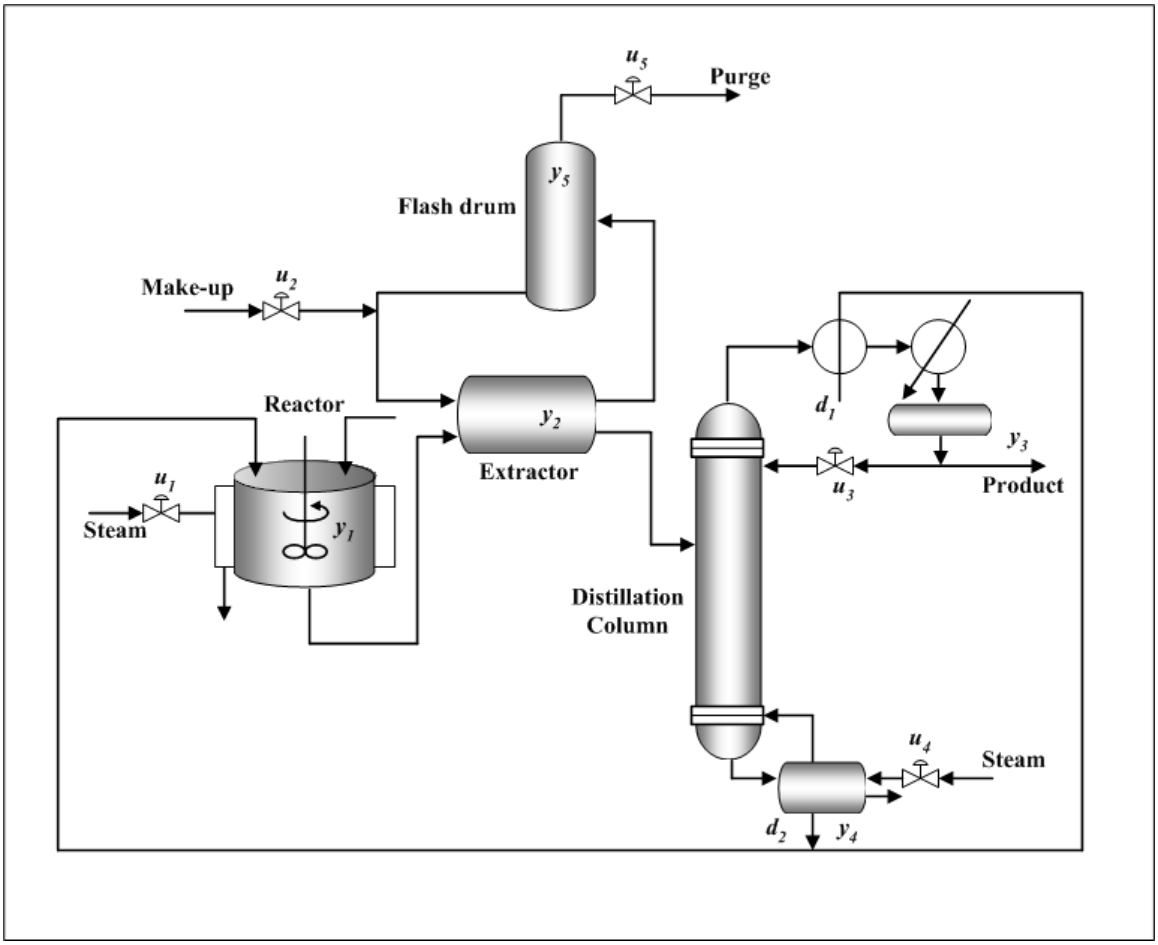


Figure 4.6 Reactor/separator system (Lee *et al.* 2000)

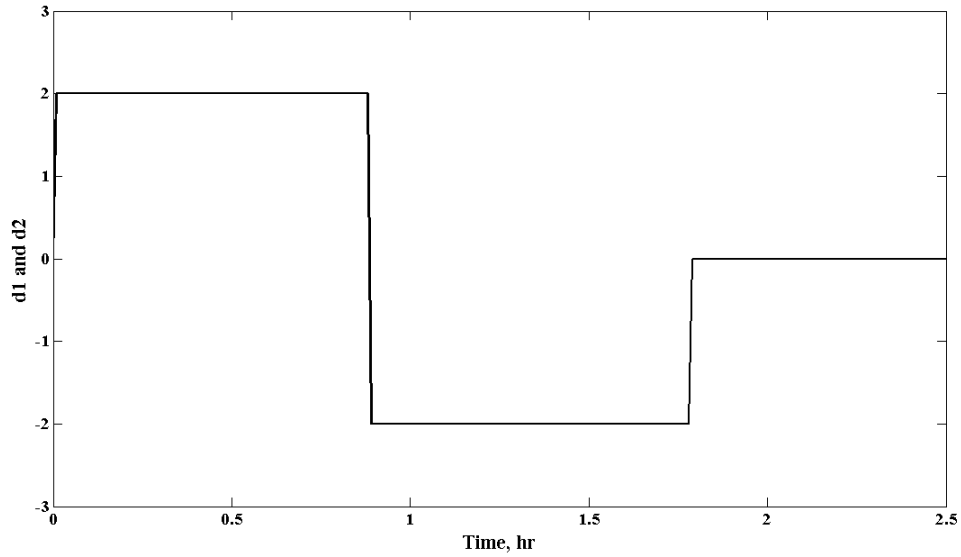


Figure 4.7 Simulated disturbances d_1 and d_2.

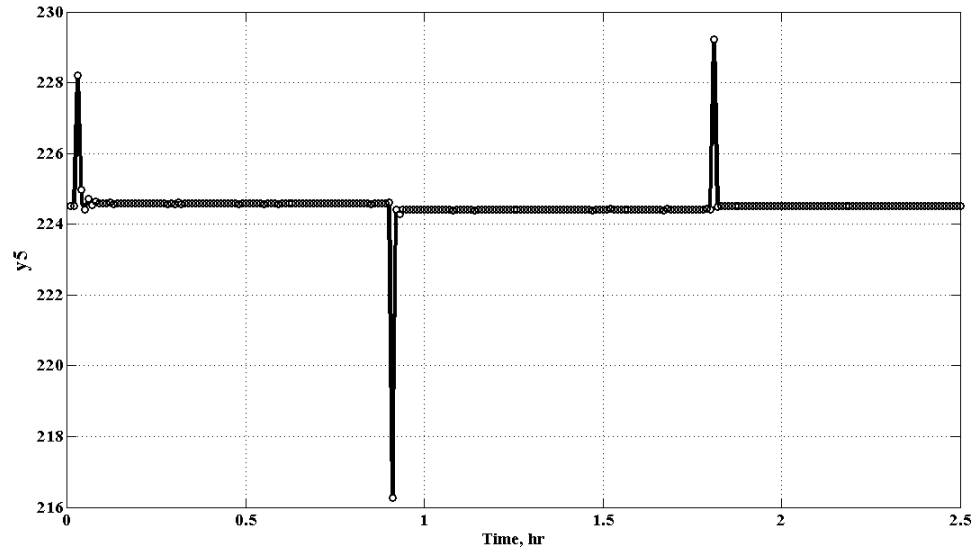


Figure 4.8 Dynamic response of y_5: Centralized (solid line), RDMPC (circles).

Table 4.4 Inputs and outputs of the process

Controlled outputs	Manipulated inputs	Input disturbances
y_1, reactor temperature	u_1, steam flow rate	d_1, flowrate to the reactor
y_2, raffinate composition	u_2, make-up flowrate	d_2, bottom flowrate
y_3, product composition	u_3, reflux flowrate	
y_4, bottoms composition	u_4, boil-up rate	
y_5, flash-drum pressure	u_5, purge-flowrate	

4.5 Conclusions

The main goal of this work was to propose an on-line algorithm for RDMPC strategy that explicitly considers model errors. The key idea of the proposed method is to decompose the model of the whole system into N subsystems and then obtain a local state feedback controller by minimizing an upper bound on a robust performance objective for each subsystem. The subsystem performance takes into account the objectives of the other subsystems in order to achieve the goal of the entire system. The method was also suitable for pursuing other control objectives such as Nash equilibrium or decentralized control. The problem was converted into N convex problems with linear matrix inequalities and solved iteratively by using the Jacobi iteration method with successive relaxation (SR). Although convergence of the iterative solution was proven, the SR feature was helpful for filtering numerical noise in the LMI solutions resulting in faster convergence. When convergence was reached, the algorithm led to the same solution of the centralized MPC problem. In addition, the algorithm was extended for output feedback by including an observer. It was also proven that if the algorithm is terminated

at any feasible intermediate iteration the robust stability is still maintained. The examples showed that RDMPC can achieve, after a sufficient number of iterations, similar performance to centralized control. Moreover, the examples illustrated that improvements in RDMPC performance as compared to decentralized and Nash control can be achieved with a relatively small number of iterations. The RDMPC algorithm was also shown that when it is terminated before reaching convergence it can provide lower computation time compared to Centralized MPC especially while the loss in performance is not significant.

CHAPTER 5

A Closed-Loop Dual-Mode Approach for Robust Distributed Model Predictive Control

5.1 Overview

This chapter proposes a new robust distributed model predictive control framework that uses a closed-loop dual-mode approach to reduce the demanding computations required to satisfy robust constraints. The proposed algorithm requires solving N convex optimization problems in parallel by allowing exchange of information among the controllers. A relaxation technique is also developed to overcome the problem of feasibility for the initial iteration. Two simulation examples are used to illustrate the new method and for comparison with a previously developed technique in terms of performance and maximum CPU time per control interval. The simulation results showed that the new algorithm provides a significant reduction in online computations.

5.2 Introduction

Model predictive control (MPC) has been successfully applied in the process industry for the last 2 decades (Qin and Badgwell 2003). The success of MPC stems from its ability to control multivariable process systems and to explicitly handling constraints on process variables. Recently, distributed MPC (DMPC) architectures have received great attention motivated by their advantage for providing similar performance to

centralized MPC while maintaining flexibility against failures and partial shut-downs due to its decentralized nature. The main idea in DMPC is to allow the different MPCs to communicate and be coordinated so as to achieve full cooperation by considering the overall system objective in each controller (Rawlings *et al.* 2008, Zhang and Li 2007). The special case of Nash-equilibrium may take place when each controller has either a strict local objective or in addition to the local objective the other objectives are considered with less priority (Li *et al.* 2005). Rawlings *et al.* (2008) and Scattolini (2009) presented recent reviews and further insights on DMPC.

The coordination strategies reported in the literature employ linear models to predict the future behavior of the process in order to achieve the optimal or sub-optimal closed-loop performance and they rely on feedback to correct for any model uncertainty. However, in reality, linear models are never accurate due to nonlinearity or inaccurate identification and when model errors are severe feedback-based corrections may not be enough. The robustness of distributed control strategies to plant-model mismatch has been identified as one of the major factors for the successful application of DMPC strategies (Rawlings *et al.* 2008). Methods for design of DMPC algorithms that are robust with respect to model errors received little attention in the literature regardless of the rich theory in robust centralized MPC. In a recent work, Al-Gherwi *et al.* (2009, 2010) proposed a new algorithm for robust DMPC, referred heretofore as RDMPC1, which deals explicitly with plant-model mismatch. In this method the LMI-based robust MPC formulation proposed by Kothare *et al.* (1996) has been modified into an iterative algorithm for RDMPC. The algorithm requires decomposing the entire system into N subsystems and solving N convex optimization problems to minimize an upper bound on

a robust performance objective by using a time-varying state-feedback controller for each subsystem. The algorithm has been shown to provide stability and its performance becomes progressively similar to centralized MPC as the iterative algorithm reaches full convergence. However, the iterative nature of the algorithm and the requirement to consider large robust stability constraints has been found to increase the online computations. Therefore the main goal of this work is to present a new methodology to overcome the online computational problems.

Kouvaritakis *et al.* (2000) developed a closed-loop dual-mode paradigm in which the control law is parameterized with a fixed state feedback computed offline and additional degrees of freedom that appear in the first *Nc* control moves to be computed online. The function of the additional degree of freedom is to steer the states when the constraints are active into an invariant set corresponding to an unconstrained state feedback that is computed offline. Therefore, most of the heavy computations can be handled offline in which invariance and feasibility constraints are satisfied rendering online computations more efficient since it requires satisfying smaller constraints. The methodology presented in the current work extends upon the result of Kouvaritakis *et al.* (2000) providing a new framework for RDMPC. This proposed method will be referred heretofore as RDMPC2. The method consists in decomposing the centralized MPC control problem into *N* subproblems where each subproblem solves an MPC controller for a particular subsystem. Then, the controllers are coordinated among themselves online via exchange of information in order to ensure feasibility. Since RDMPC2 is iterative and the invariant set is computed offline then a feasible initial guess is essential for the algorithm to start and satisfy constraints. It is proposed in the current method to

introduce slack variables and develop a relaxation method to guarantee a feasible initial start. While the use of slack variables allow for slight constraints violation at the initial iteration, these variables approach zero values in the next iterations and the constraints are respected afterwards. The proposed method is shown to guarantee robust stability and feasibility. An additional feature of the proposed methodology is that it allows for different control objectives to be optimized. For example, the formulations are shown for a Nash optimization objective is presented. Finally the proposed algorithm is compared with the RDMPC1 algorithm presented in a previous chapter. This work is organized as follows: In section 5.3 the previous RDMPC1 algorithm is reviewed after then the new proposed algorithm is introduced in section 5.4. Application examples and comparisons are given in section 5.5. Section 5.6 concludes the chapter.

5.3 Review of RDMPC1 Algorithm

In this section the algorithm proposed by Al-Gherwi *et al.* (2009) is reviewed. The objective is to control the following linear time varying system:

$$\boldsymbol{x}(k+1) = \boldsymbol{A}(k)\boldsymbol{x}(k) + \boldsymbol{B}(k)\boldsymbol{u}(k) \tag{5.1}$$

$$[\boldsymbol{A}(k)\ \boldsymbol{B}(k)] = \sum_{l=1}^{L} \beta_l [\boldsymbol{A}^{(l)}\ \boldsymbol{B}^{(l)}] \quad ; \sum_{l=1}^{L} \beta_l = 1;\ \beta_l \geq 0 \tag{5.2}$$

where $\boldsymbol{x} \in \Re^n$; $\boldsymbol{u} \in \Re^m$ are the process states and inputs; respectively. The actual plant is represented by the polytopic description given in (5.2). Then (5.1) and (5.2) can be decomposed into *N* subsystems as:

$$\boldsymbol{x}_i(k+1) = \boldsymbol{A}_i(k)\boldsymbol{x}_i(k) + \boldsymbol{B}_i(k)\boldsymbol{u}_i(k) + \sum_{\substack{j=1 \\ j\neq i}}^{N} \boldsymbol{B}_j(k)\boldsymbol{u}_j(k) \tag{5.3}$$

$$[\boldsymbol{A}_i(k)\ \boldsymbol{B}_i(k)..\ \boldsymbol{B}_j(k)..] = \sum_{l=1}^{L} \beta_l [\boldsymbol{A}_i^{(l)}\ \boldsymbol{B}_i^{(l)}..\ \boldsymbol{B}_j^{(l)}..] \quad \forall i \in \{1,...,N\},\ j \neq i \tag{5.4}$$

where $\boldsymbol{x}_i' = \left[\boldsymbol{x}_{11}', \cdots\cdots, \boldsymbol{x}_{ii}', \cdots\cdots, \boldsymbol{x}_{NN}'\right]'$ is the vector of states of subsystem i containing states $\boldsymbol{x}_{ii}$ that can be measured or estimated locally augmented with states $\boldsymbol{x}_{jj}$ that are measured or estimated by the other subsystems and are exchanged via communication. The $\boldsymbol{A}_i(k)$ contains all the elements of the matrix $\boldsymbol{A}(k)$ in (5.1). Then every subsystem i solves the following min-max problem:

$$\begin{aligned} &\min_{\boldsymbol{u}_i(k+n|k)} \ \max_{[\boldsymbol{A}_i(k+n)\ \boldsymbol{B}_i(k+n)\ \boldsymbol{B}_j(k+n)],n\geq 0} \ J_i(k) \\ &s.t. \\ &\qquad \left|\boldsymbol{u}_i(k+n|k)\right| \leq \boldsymbol{u}_i^{max},\ n \geq 0 \end{aligned} \tag{5.5}$$

The local objective $J_i(k)$ is defined as:

$$\begin{aligned} J_i(k) = \sum_{n=0}^{\infty} [&\boldsymbol{x}_i'(k+n|k)\boldsymbol{S}_i\boldsymbol{x}_i(k+n|k) + \boldsymbol{u}_i'(k+n|k)\boldsymbol{R}_i\boldsymbol{u}_i(k+n|k) \\ &+ \sum_{\substack{i=1 \\ i\neq j}}^{N} \boldsymbol{u}_j^{\bullet\prime}(k+n|k)\boldsymbol{R}_j\boldsymbol{u}_j^{\bullet}(k+n|k)] \end{aligned} \tag{5.6}$$

where $\boldsymbol{S}_i > 0$, $\boldsymbol{R}_i > 0$, $\boldsymbol{R}_j > 0$. The local objective given in (5.6) takes into account the goals of the other controllers, third summation in the RHS, in order to achieve the global

objective of the entire system. The superscript "•" indicates that the solution was obtained in a previous iteration and remains fixed in the current iteration. The objective given in (5.6) can take different formulations; namely; cooperative and Nash. In the latter, strictly local objectives are used where only local states and inputs are considered. Instead of solving the min-max problem (5.5), it is replaced by solving the following convex problem where an upper bound γ_i on $J_i(k)$ is minimized:

$$
\begin{aligned}
&\min_{\gamma_i, \boldsymbol{Q}_i, \boldsymbol{Y}_i} \ \gamma_i \\
&s.t. \\
&\begin{bmatrix} 1 & \boldsymbol{x}_i'(k) \\ \boldsymbol{x}_i(k) & \boldsymbol{Q}_i \end{bmatrix} \geq 0 \\
&\begin{bmatrix} \boldsymbol{Q}_i & \boldsymbol{Q}_i \tilde{\boldsymbol{A}}_i'^{(l)} + \boldsymbol{Y}_i' \boldsymbol{B}_i'^{(l)} & \boldsymbol{Q}_i \tilde{\boldsymbol{S}}_i^{1/2} & \boldsymbol{Y}_i' \boldsymbol{R}_i^{1/2} \\ * & \boldsymbol{Q}_i & \boldsymbol{0} & \boldsymbol{0} \\ * & * & \gamma_i \boldsymbol{I} & \boldsymbol{0} \\ * & * & * & \gamma_i \boldsymbol{I} \end{bmatrix} \geq 0 \\
&\qquad\qquad \forall l \in \{1, \ldots, L\} \\
&\begin{bmatrix} \left(\boldsymbol{u}_i^{max}\right)^2 \boldsymbol{I} & \boldsymbol{Y}_i \\ \boldsymbol{Y}_i' & \boldsymbol{Q}_i \end{bmatrix} \geq 0
\end{aligned}
\tag{5.7}
$$

Where $\boldsymbol{F}_i = \boldsymbol{Y}_i' \boldsymbol{Q}_i^{-1}$, $\boldsymbol{Q}_i = \gamma_i \boldsymbol{P}_i^{-1}$, $\tilde{A}_i(k) = A_i(k) + \sum_{\substack{j=1 \\ j \neq i}}^{N} B_j(k) F_j^{\bullet}(k)$, and

$$\tilde{\boldsymbol{S}}_i = \boldsymbol{S}_i + \sum_{\substack{i=1 \\ i \neq j}}^{N} \boldsymbol{F}_j^{\bullet'}(k+n|k) \boldsymbol{R}_j \boldsymbol{F}_j^{\bullet}(k+n|k).$$

Then RDMPC1 algorithm can be implemented online as illustrated below:

RDMPC1 algorithm

***Step0 (initialization)**: at control interval k=0 set $\boldsymbol{F}_i$=0.*

***Step1 (updating)** at the beginning of control interval (k) all the controllers exchange their local states measurements and initial estimates $\boldsymbol{F}_i$'s via communication, set iteration t = 0 and $\boldsymbol{F}_i = \boldsymbol{F}_i^{(0)}$.*

Step2 (iterations)

while $t \le t_{max}$

Solve all N LMI problems (5.7) in parallel to obtain the minimizers $\boldsymbol{Y}_i^{(t)}, \boldsymbol{Q}_i^{(t)}$ to estimate the feedback solutions $\boldsymbol{F}_i^{(t)} = \boldsymbol{Y}_i'^{(t)} \boldsymbol{Q}_i^{-1(t)}$. If problem is infeasible set $\boldsymbol{F}_i^{(t)} = \boldsymbol{F}_i^{(t-1)}$. Check the convergence for a specified error tolerance ε_i for all the controllers

if $\left\| \boldsymbol{F}_i^{(t)} - \boldsymbol{F}_i^{(t-1)} \right\| \le \varepsilon_i \quad \forall i \in \{1,...,N\}$

break

end if

Exchange the solutions $\boldsymbol{F}_i$'s and set t = t + 1

end while

***Step3 (implementation)** apply the control actions $\boldsymbol{u}_i = \boldsymbol{F}_i \boldsymbol{x}_i$ to the corresponding subsystems, increase the control interval k = k + 1, return to step1 and repeat the procedure.*

Theorem 1. *At sampling time k and any iteration t > 0, the state feedback solutions $\boldsymbol{F}_i^{(t)}(k) = \boldsymbol{Y}_i'^{(t)}(k) \boldsymbol{Q}_i^{-1(t)}(k)$, $i \in \{1,\cdots,N\}$, obtained from Algorithm 1, robustly*

asymptotically stabilize the closed loop system $\boldsymbol{x}(k+1)=\left(\boldsymbol{A}(k)+\sum_{i=1}^{N}\boldsymbol{B}_i(k)\boldsymbol{F}_i^{(t)}(k)\right)\boldsymbol{x}(k)$

where $\boldsymbol{A}(k)$ *and* $\boldsymbol{B}_i(k)$ *belong to the polytopic description defined in (5.4)* .

Proof. The proof is given in Chapter 4.

It should be pointed out here that RDMPC1 algorithm can achieve the robust centralized performance when the cooperative scheme is considered. This is not the case when Nash-based objective is used since local objectives are considered. It has also been shown that robust stability can be satisfied even when the algorithm is terminated before convergence if all the *N* problems given by (5.7) are feasible. In the previous chapter it has been reported that increasing iterations can improve performance and become close to that of centralized. However, this could also increase the computation time requirements since problem (5.7) contains robust stability and feasibility constraints that must be satisfied online. Therefore reducing the computation time is essential in order to facilitate the application of the algorithm in real-time applications. In order to achieve this goal, a new framework is proposed in the next section.

5.4 New RDMPC framework (RDMPC2 Algorithm)

In the RDMPC1 algorithm proposed in the previous chapter, the controllers compute their $\boldsymbol{F}_i(k)$ iteratively at each control interval by minimizing an upper bound on the objective function while satisfying constraints. It has been shown that choosing to terminate the algorithm at any iteration still ensures robust stability. However, because of the iterative nature of the algorithm where problem (5.7) is solved repeatedly with

possibly large number of constraints the online computations can become very expensive. In order to overcome this problem in this section it is proposed to use the closed-loop dual-mode paradigm (Kouvaritakis *et al.* 2000) to tackle the distributed MPC problem. In this approach the control law is parameterized by including an additional degree of freedom as follows:

$$\boldsymbol{u}_i(k) = \begin{cases} \boldsymbol{F}_i \boldsymbol{x}_i(k) + \boldsymbol{c}_i(k) & k = 0,\ldots,Nc \\ \boldsymbol{F}_i \boldsymbol{x}_i(k) & k \geq Nc \end{cases} \tag{5.8}$$

where $\boldsymbol{F}_i$ is now a fixed gain to be computed offline, $\boldsymbol{c}_i(k)$ are the additional degrees of freedom to be computed online, and Nc is the control horizon. To obtain the closed-loop system, the control law (5.8) is used together with the uncertainty description (5.3) and (5.4) and after some straightforward algebraic manipulations the state-space closed-loop model is obtained as follows:

$$z(k+1) = \boldsymbol{\Psi}(k) z(k) \tag{5.9}$$

where $z(k) = \begin{bmatrix} \boldsymbol{x}(k) \\ \boldsymbol{f}(k) \end{bmatrix}$, $\boldsymbol{f}(k) = \begin{bmatrix} \boldsymbol{f}_1(k) \\ \vdots \\ \boldsymbol{f}_i(k) \\ \vdots \\ \boldsymbol{f}_N(k) \end{bmatrix}$, $\boldsymbol{f}_i(k) = \begin{bmatrix} \boldsymbol{c}_i(k) \\ \boldsymbol{c}_i(k+1) \\ \vdots \\ \vdots \\ \boldsymbol{c}_i(k+Nc-1) \end{bmatrix}$,

$$\boldsymbol{\Psi}(k)=\begin{bmatrix} \boldsymbol{\Phi}(k) & \begin{bmatrix}\boldsymbol{B}_1(k) & \boldsymbol{0}_{nx,(Nc-1)nu_1}\end{bmatrix} & \cdots & \begin{bmatrix}\boldsymbol{B}_i(k) & \boldsymbol{0}_{nx,(Nc-1)nu_i}\end{bmatrix} & \cdots & \begin{bmatrix}\boldsymbol{B}_N(k) & \boldsymbol{0}_{nx,(Nc-1)nu_N}\end{bmatrix} \\ \boldsymbol{0}_{(Nc)nu_1,nx} & \boldsymbol{M}_1 & \boldsymbol{0}_{(Nc)nu_1,nx} & \cdots & \cdots & \boldsymbol{0}_{(Nc)nu_1,nx} \\ \vdots & \ddots & \ddots & & & \\ \boldsymbol{0}_{(Nc)nu_i,nx} & \cdots & \boldsymbol{0}_{(Nc)nu_i,nx} & \boldsymbol{M}_i & \cdots & \boldsymbol{0}_{(Nc)nu_i,nx} \\ \vdots & & & & \ddots & \\ \boldsymbol{0}_{(Nc)nu_N,nx} & \cdots & \cdots & \cdots & \cdots & \boldsymbol{M}_N \end{bmatrix}$$

$\boldsymbol{\Phi}(k)=\boldsymbol{A}(k)+\boldsymbol{B}(k)\boldsymbol{F}$, $\boldsymbol{F}=\begin{bmatrix}\boldsymbol{F}_1\\ \vdots \\ \boldsymbol{F}_i \\ \vdots \\ \boldsymbol{F}_N\end{bmatrix}$, $\boldsymbol{M}_i=\begin{bmatrix}\boldsymbol{0}_{(Nc-1)nu_i,nu_i} & \boldsymbol{I}_{(Nc-1)nu_i} \\ \boldsymbol{0}_{nu_i,nu_i} & \boldsymbol{0}_{nu_i,(Nc-1)nu_i}\end{bmatrix}$, nx is the number of total states, nu_i is the number of inputs of subsystem i, $i=1,\ldots,N$.

The closed-loop given above is slightly different from the original formulation by (Kouvaritakis *et al.* 2000) in that the system is rearranged in such away to allow distributed computations as to be explained later.

5.4.1 Offline computations

The idea is to design unconstrained gains $\boldsymbol{F}_i$ and perform offline optimization to enlarge the feasible region defined by the invariant set $\varepsilon_z=\left\{z\in\Re^n \,\middle|\, z'\boldsymbol{Q}_z^{-1}z\le 1\right\}$. The problem of maximizing the invariant set can be formulated as the following convex optimization problem (Boyd et al. 1994):

$$\min_{Q_z} \ \log\det\left(\begin{bmatrix} \boldsymbol{I}_{nx} & \boldsymbol{0}_{nx,Nc\sum nu_i} \end{bmatrix} \boldsymbol{Q}_z \begin{bmatrix} \boldsymbol{I}_{nx} & \boldsymbol{0}_{nx,Nc\sum nu_i} \end{bmatrix}'\right)^{-1} \tag{5.10}$$

s.t.

$$\begin{bmatrix} \boldsymbol{Q}_z & \boldsymbol{Q}_z\boldsymbol{\Psi}^{(l)\prime} \\ \boldsymbol{\Psi}^{(l)}\boldsymbol{Q}_z & \boldsymbol{Q}_z \end{bmatrix} \geq 0, \ \forall l \in \{1,...,L\}$$

$$\left(\boldsymbol{u}_i^{max}\right)^2 - \begin{bmatrix} \boldsymbol{F}_i & \boldsymbol{I}_{nu_i} & \boldsymbol{0}_{nu_i,(Nc-1)\sum nu_i} \end{bmatrix} \boldsymbol{Q}_z \begin{bmatrix} \boldsymbol{F}_i & \boldsymbol{I}_{nu_i} & \boldsymbol{0}_{nu_i,(Nc-1)\sum nu_i} \end{bmatrix}' \geq 0$$

The first and second constraints represent invariance and feasibility; respectively. The invariance constraint forces the states to evolve within an invariant set. Problem (5.10) can be solved using YALMIP interface (Lofberg 2004) integrated with MATLAB® LMI solvers.

At this point the offline computations are explained. In this work two distributed schemes are considered: cooperative and Nash-based objective. In the cooperative scheme every local objective takes into account the optimization objective of the entire system which is identical to the objective used for centralized MPC thus the algorithm can achieve centralized performance following convergence. On the other hand, in the Nash-based scheme every subsystem considers strictly local objective. The gains $\boldsymbol{F}_i$ in the cooperative scheme are obtained by partitioning an unconstrained centralized gain $\boldsymbol{F}$. The following centralized problem is solved offline to obtain $\boldsymbol{F} = \boldsymbol{Y}'\boldsymbol{Q}^{-1}$:

$$\min_{\gamma,\boldsymbol{Q},\boldsymbol{Y}} \ \gamma$$

$$s.t. \quad \begin{bmatrix} \boldsymbol{Q} & \boldsymbol{Q}\boldsymbol{A}'^{(l)} + \boldsymbol{Y}'\boldsymbol{B}'^{(l)} & \boldsymbol{Q}\,\boldsymbol{S}^{1/2} & \boldsymbol{Y}'\boldsymbol{R}^{1/2} \\ * & \boldsymbol{Q} & \boldsymbol{0} & \boldsymbol{0} \\ * & * & \gamma\boldsymbol{I} & \boldsymbol{0} \\ * & * & * & \gamma\boldsymbol{I} \end{bmatrix} \geq 0 \qquad (5.11)$$

$$\forall l \in \{1,\ldots,L\}$$

The $\boldsymbol{F}_i$ in the case of Nash are solved by solving the following problem iteratively:

$$\min_{\gamma_i,\boldsymbol{Q}_i,\boldsymbol{Y}_i} \ \gamma_i$$

$$s.t. \quad \begin{bmatrix} \boldsymbol{Q}_i & \boldsymbol{Q}_i\tilde{\boldsymbol{A}}_i'^{(l)} + \boldsymbol{Y}_i'\boldsymbol{B}_i'^{(l)} & \boldsymbol{Q}_i\tilde{\boldsymbol{S}}_i^{1/2} & \boldsymbol{Y}_i'\boldsymbol{R}_i^{1/2} \\ * & \boldsymbol{Q}_i & \boldsymbol{0} & \boldsymbol{0} \\ * & * & \gamma_i\boldsymbol{I} & \boldsymbol{0} \\ * & * & * & \gamma_i\boldsymbol{I} \end{bmatrix} \geq 0 \qquad (5.12)$$

$$\forall l \in \{1,\ldots,L\}$$

The formulation of the Nash equilibrium based scheme is similar to that reported previously in chapter 4.

The summary of offline computations are as follows:

1) Obtain the unconstrained gains $\boldsymbol{F}_i$ for either cooperative or Nash schemes using equations (5.11) or (5.12); respectively.
2) Choose the control horizon Nc and enlarge the invariant set by solving problem (5.10).

5.4.2 Online computations

Since the gains $\boldsymbol{F}_i$ are obtained offline either from (5.11) or (5.12) by minimizing an upper bound on the objective function then the online optimization consists in minimizing the additional degrees of freedom $\boldsymbol{f}_i(k)$ which are treated as an external

perturbations to the corresponding subsystems (Kouvaritakis *et al.* 2000). Therefore, the online optimization problem for subsystem i, $i = 1,\ldots,N$, when the cooperative scheme is implemented is as follows:

$$\min_{f_i(k),\lambda_i} \boldsymbol{f}_1^{(t-1)}(k)' \boldsymbol{W}_{f_1} \boldsymbol{f}_1^{(t-1)}(k) + \cdots + \boldsymbol{f}_i^{(t)}(k)' \boldsymbol{W}_{f_i} \boldsymbol{f}_i^{(t)}(k) + \cdots + \boldsymbol{f}_N^{(t-1)}(k)' \boldsymbol{W}_{f_N} \boldsymbol{f}_N^{(t-1)}(k) + \omega \lambda_i^{(t)2}$$

s.t.

$$\boldsymbol{z}_i^{(t)'}(k) \boldsymbol{Q}_z^{-1} \boldsymbol{z}_i^{(t)}(k) \le 1 + \lambda_i^{(t)} \tag{5.13}$$

where t is the iteration number, $\boldsymbol{W}_{f_i}$ is a weighting matrix, λ_i is a scalar slack variable, ω is a penalty factor, and

$$\boldsymbol{z}_i(k) = \begin{bmatrix} \boldsymbol{x}(k) \\ \boldsymbol{f}(k) \end{bmatrix}, \boldsymbol{f}(k) = \begin{bmatrix} \boldsymbol{f}_1^{(t-1)}(k) \\ \vdots \\ \boldsymbol{f}_i^{(t)}(k) \\ \vdots \\ \boldsymbol{f}_N^{(t-1)}(k) \end{bmatrix}.$$

The vector of states $\boldsymbol{x}(k)$ contains the local estimated or measured states augmented with the states received via communication from the other subsystems at the beginning of control interval. The invariance constraint is obtained offline and because of the iterative nature of the proposed method a feasible solution may not be easily obtained initially. One possible way is to search for a feasible initial guess that satisfy the constraint but that may not be practical to be implemented online since it has been found to be a time consuming step. Therefore, a slack variable λ_i is introduced in each subsystem's problem to allow for an initial small violation of the constraint whereas this variable is penalized

so as to force it to decrease very rapidly. Since the problem is convex then the slack variables approach zero immediately at early iterations. By applying the Schur complement to the objective function and the constraint and by dropping the sample time (k) for ease of notation, problem (5.13) can be transformed into the following LMI problem:

$$\min_{\alpha_i^{(t)},\lambda_i^{(t)},\boldsymbol{f}_i^{(t)}} \alpha_i^{(t)} \tag{5.14}$$

s.t.

$$\begin{bmatrix} \alpha_i^{(t)} & \boldsymbol{f}_1^{(t-1)\prime} & \cdots & \boldsymbol{f}_i^{(t)\prime} & \cdots & \boldsymbol{f}_N^{(t-1)\prime} & \lambda_i^{(t)} \\ * & \boldsymbol{W}_{f_1} & \boldsymbol{0} & & \vdots & & \boldsymbol{0} \\ \vdots & \boldsymbol{0} & \ddots & & & & \\ * & \vdots & \ddots & \boldsymbol{W}_{f_i} & & & \vdots \\ \vdots & & & & \ddots & & \\ * & \vdots & & & \ddots & \boldsymbol{W}_{f_N} & \boldsymbol{0} \\ * & \boldsymbol{0} & & \cdots & & \boldsymbol{0} & \omega \end{bmatrix} \geq 0$$

$$\begin{bmatrix} 1+\lambda_i^{(t)} & \boldsymbol{x}' & \boldsymbol{f}_1^{(t-1)\prime} & \cdots & \boldsymbol{f}_i^{(t)\prime} & \cdots & \boldsymbol{f}_N^{(t-1)\prime} \\ * & \boldsymbol{Q}_{xx} & \boldsymbol{Q}_{f_1 x} & \cdots & \boldsymbol{Q}_{f_i x} & \cdots & \boldsymbol{Q}_{f_N x} \\ * & * & \boldsymbol{Q}_{f_1} & \cdots & \boldsymbol{Q}_{f_1 f_i} & \cdots & \boldsymbol{Q}_{f_1 f_N} \\ \vdots & \vdots & & \ddots & & \ddots & \\ * & * & & & \boldsymbol{Q}_{f_i} & & \boldsymbol{Q}_{f_i f_N} \\ \vdots & \vdots & & & & \ddots & \\ * & * & & \cdots & & * & \boldsymbol{Q}_{f_N} \end{bmatrix} \geq 0$$

where $\boldsymbol{Q}_{xx}, \boldsymbol{Q}_{f_i x}, \boldsymbol{Q}_{f_i}, \boldsymbol{Q}_{f_i f_j}$ result from appropriate partitioning of the original matrix $\boldsymbol{Q}_z$ that is computed offline.

In the case of Nash-based scheme, the online problem is slightly different. The objective function that would be minimized online is now $\boldsymbol{f}_i^{(t)}(k)'\boldsymbol{W}_{f_i}\boldsymbol{f}_i^{(t)}(k)+\omega\lambda_i^{(t)2}$ and the corresponding LMI problem is given by:

$$\min_{\alpha_i^{(t)},\lambda_i^{(t)},\boldsymbol{f}_i^{(t)}} \alpha_i^{(t)} \tag{5.15}$$

s.t.

$$\begin{bmatrix} \alpha_i^{(t)} & \boldsymbol{f}_i^{(t)\prime} & \lambda_i^{(t)} \\ \boldsymbol{f}_i^{(t)} & \boldsymbol{W}_{f_i} & \boldsymbol{0} \\ \lambda_i^{(t)} & \boldsymbol{0} & \omega \end{bmatrix} \geq 0$$

$$\begin{bmatrix} 1+\lambda_i^{(t)} & \boldsymbol{x}' & \boldsymbol{f}_1^{(t-1)\prime} & \cdots & \boldsymbol{f}_i^{(t)\prime} & \cdots & \boldsymbol{f}_N^{(t-1)\prime} \\ * & \boldsymbol{Q}_{xx} & \boldsymbol{Q}_{f_1x} & \cdots & \boldsymbol{Q}_{f_ix} & \cdots & \boldsymbol{Q}_{f_Nx} \\ * & * & \boldsymbol{Q}_{f_1} & \cdots & \boldsymbol{Q}_{f_1f_i} & \cdots & \boldsymbol{Q}_{f_1f_N} \\ \vdots & \vdots & & \ddots & & \ddots & \\ * & * & & & \boldsymbol{Q}_{f_i} & & \boldsymbol{Q}_{f_if_N} \\ \vdots & \vdots & & & & \ddots & \\ * & * & & \cdots & & * & \boldsymbol{Q}_{f_N} \end{bmatrix} \geq 0$$

It should be pointed out that the $\boldsymbol{Q}_z$ in Nash-based is different than that of the cooperative scheme since the latter is equivalent to the feasible region obtained by centralized control. Problems (5.14) and (5.15) are solved using the function *mincx* in MATLAB® robust control toolbox. The algorithm RDMPC2 is summarized below:

RDMPC2 algorithm

***Step0 (initialization)**: at control interval k=0 set $\boldsymbol{f}_i=\boldsymbol{0}_{(Nc-1)nui,1}$.*

***Step1 (updating)** at the beginning of control interval (k) all the controllers exchange their local states measurements and initial estimates $\boldsymbol{f}_i$'s via communication, set iteration t = 0 and $\boldsymbol{f}_i = \boldsymbol{f}_i^{(0)}$.*

Step2 (iterations)

while $t \le t_{max}$

Solve all N LMI problems either (5.14) or (5.15) in parallel to obtain the minimizers $\boldsymbol{f}_i^{(t)}, \alpha_i^{(t)}, \lambda_i^{(t)}$. Check the convergence for a specified error tolerance ε_i for all the controllers

if $\left\| \boldsymbol{f}_i^{(t)} - \boldsymbol{f}_i^{(t-1)} \right\| \le \boldsymbol{\varepsilon}_i \quad \forall i \in \{1,...,N\}$

break

end if

Exchange the solutions $\boldsymbol{f}_i$'s and set t = t + 1

end while

***Step3 (implementation)** apply the control actions $\boldsymbol{u}_i(k) = \boldsymbol{F}_i\boldsymbol{x}_i(k) + \boldsymbol{f}_i(k)$ to the corresponding subsystems, increase the control interval k = k + 1, return to step1 and repeat the procedure.*

In the RDMPC2 algorithm the solutions can be filtered to dampen out any numerical noise and improve convergence as follows:

$$\boldsymbol{f}_i^{(t+1)} = \alpha \overline{\boldsymbol{f}}_i^{(t+1)} + (1-\alpha)\boldsymbol{f}_i^{(t)} \tag{5.16}$$

where α is a parameter to be specified by the user.

Remark 1 In the case of the cooperative scheme, the RDMPC2 algorithm employs *N* identical convex problems given by (5.14) which, at convergence, are equivalent to the centralized MPC problem. The slack variables λ_i allow for an initial constraint violation. Then because of the convex feature of the problem all λ_i are decreasing and approach zero at the solution which is equivalent to centralized structure. Therefore, if the centralized solution is initially feasible then RDMPC2 is also feasible when the λ_i's approach zero. However, this is not guaranteed when the Nash scheme is implemented since the corresponding problem is no longer convex because the *N* problems given by (5.15) are not identical. Therefore, convergence in this case does depend on the existence of a Nash solution.

Assuming that RDMPC2 is feasible at initial time (k) the following theorem is stated for robust stability:

Theorem 2. *At sampling time k and* $\lambda_i \rightarrow 0$*, the corresponding control actions* $\boldsymbol{u}_i(k) = \boldsymbol{F}_i\boldsymbol{x}_i(k) + \boldsymbol{f}_i(k)$, $i \in \{1,\cdots,N\}$, *obtained from RDMPC2, robustly asymptotically stabilize the closed loop system* $\boldsymbol{z}(k+1) = \boldsymbol{\Psi}(k)\boldsymbol{z}(k)$*).*

Proof. If there is a feasible solution at time (k) then the evolution of the solutions $\boldsymbol{f}_i(k)$ in time converge to zero from $\boldsymbol{f}_i(k+1) = \boldsymbol{M}_i\boldsymbol{f}_i(k)$ and stability is ensured following the

satisfaction of constraint and the fact that the $\boldsymbol{f}_i(k)$ keep the states within the corresponding invariant set which is equivalent to stability.

5.4.3 RDMPC2 Algorithm with Output Feedback

In the previous chapter an observer design was proposed for RDMPC1. Similarly, in this section a method for RDMPC2 output feedback is proposed. The states $\boldsymbol{x}(k)$ in RDMPC2 are replaced by their estimates denoted as $\hat{\boldsymbol{x}}(k)$. The observer is designed based on a nominal model of the system that corresponds to the state space model parameters at the center of the polytopic description given in (4.2). Then, an observer is defined for each subsystem that receives all the output measurements and control actions from the other subsystems at interval k in order to perform state estimation. This estimation is conducted according to the following observer equation:

$$\hat{\boldsymbol{x}}(k+1)=\left(\boldsymbol{A}+\sum_{i=1}^{N}\boldsymbol{B}_i\boldsymbol{F}_i\right)\hat{\boldsymbol{x}}(k)+\boldsymbol{K}\left(\boldsymbol{y}(k)-\boldsymbol{C}\hat{\boldsymbol{x}}(k)\right)+\sum_{i=1}^{N}\boldsymbol{B}_i\boldsymbol{f}_i(k) \tag{5.17}$$

The observer gain $\boldsymbol{K}$ is chosen such that ($\boldsymbol{A}$-$\boldsymbol{KC}$) is stable and it is designed by a similar procedure explained in Chapter 4. Contrary to RDMPC1 where the stability of observer and controller has to be checked online, in RDMPC2 the stability can be checked offline assuming a feasible online solution does exist due to the fact that $\boldsymbol{F}_i$ were computed offline and $\boldsymbol{f}_i(k)$ are vanishing with time.

5.5 Case Studies

5.5.1 Example 1

Example 2 of chapter 4 is used to illustrate the method and for comparing the performance of RDMPC1 proposed in the previous chapter with the RDMPC2 presented in the current one. The specifications of RDMPC1 cooperative scheme used in the previous chapter are also used in this case study. For RDMPC2 cooperative scheme the following parameters are used: The unconstrained gains $\boldsymbol{F}_i$ are designed offline using equation (5.11) with $\boldsymbol{S}_{y1} = \boldsymbol{S}_{y2} = 50$, $\boldsymbol{R}_1=\boldsymbol{R}_2=15$ and the invariant set is maximized using Nc = 20. Here the unconstrained controllers had to be detuned to reduce the control horizon. For the online computations the penalty factor ω = 1e8, and filter factor of equation (5.16) α = 0.7 are used. For RDMPC2 with Nash scheme the unconstrained gains have to be detuned further to satisfy online feasibility and the weights $\boldsymbol{R}_1=\boldsymbol{R}_2=20$ are used whereas the remaining parameters are the same as the ones used in the cooperative scheme. The further detuning required for the Nash based controller is due to the strong interactions posed when the bad pairings are chosen. Such detuning was not required in RDMPC1 since the gain of the controller is tuned online.. Figures 5.1 and 5.2 show the dynamic response of the outputs for a set point change of y1s= -1 and y2s = 1 using cooperative schemes in both algorithms. The corresponding control actions are shown in Figures 5.2 and 5.3. The response using both algorithms is comparable with slightly sluggish response in RDMPC2 because of the detuning.

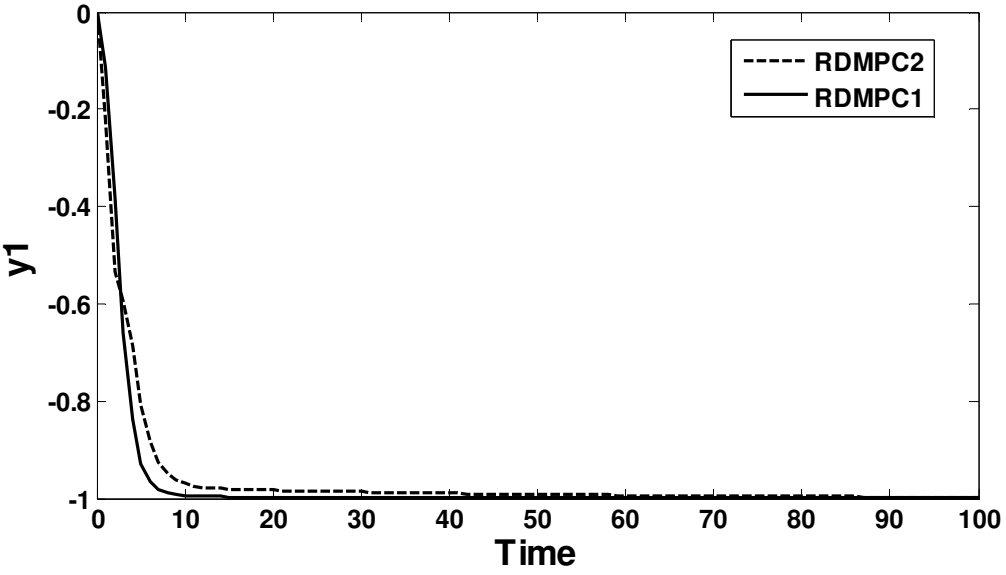


Figure 5.1 Dynamic response of y1.

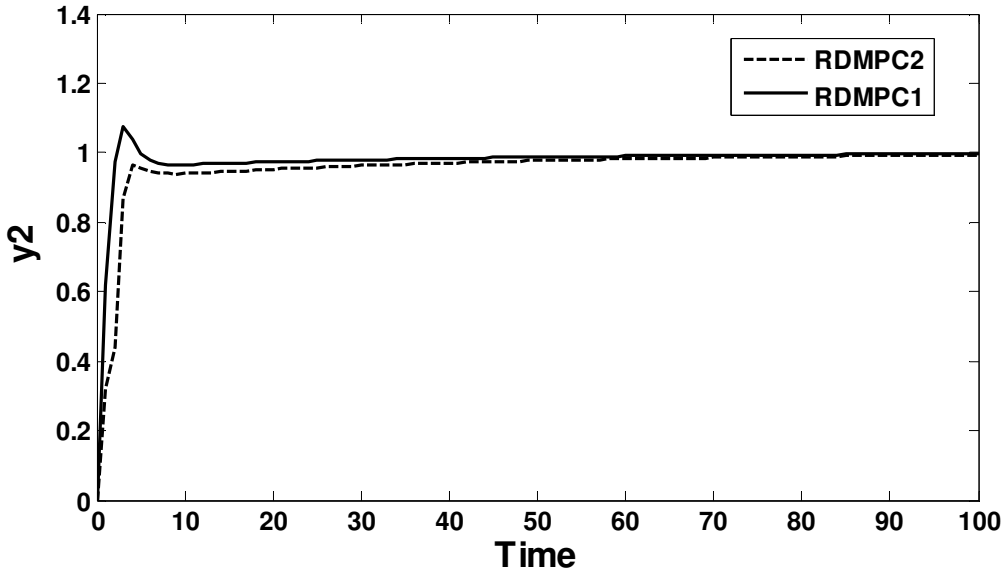


Figure 5.2 Dynamic response of y2.

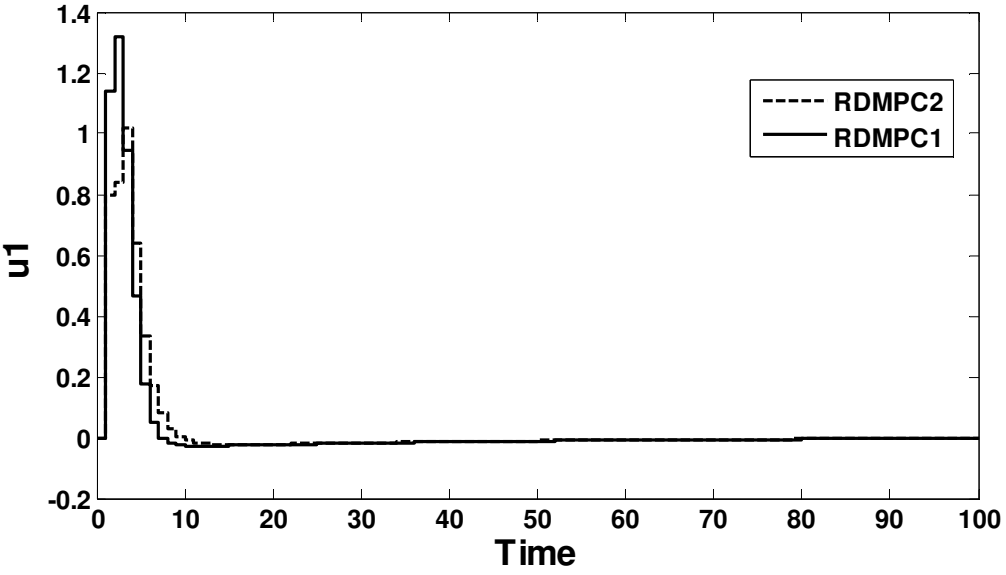


Figure 5.3 control action u1.

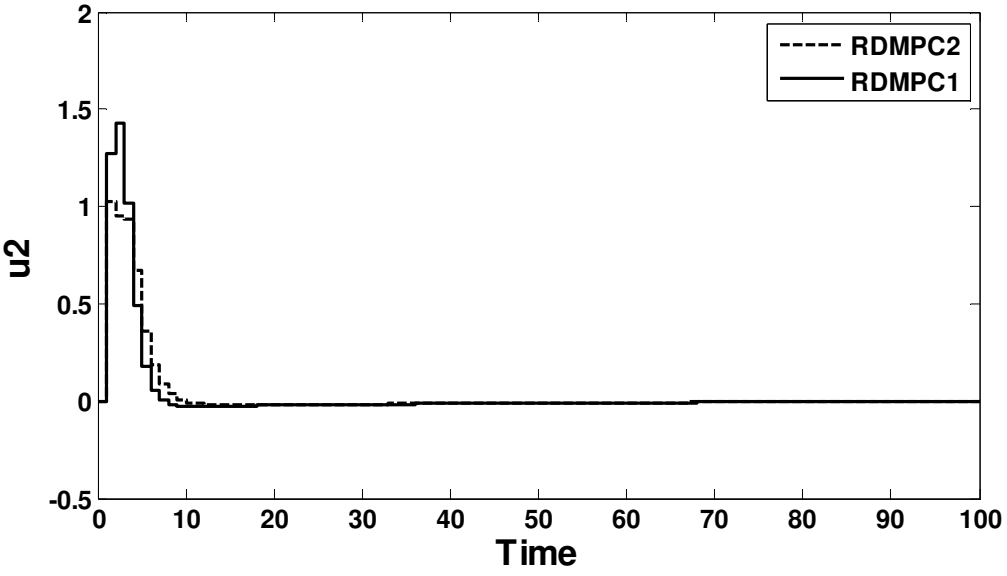


Figure 5.4 control action u2.

Figures 5.5 and 5.6 show the response when Nash scheme is implemented in RDMPC2 which results in a very sluggish response since the controllers had to be detuned to maintain feasibility. To show the convergence behavior of the RDMPC2 algorithm with cooperative scheme 100 random initial solutions were generated using a Gaussian distribution $N(0,.01)$. All the initial guesses converged to the centralized solution with maximum CPU time of 4 seconds and the λ_i converged to zero immediately in the second iterations. Figure (5.7) shows one of these cases when the algorithm converged to centralized scheme after 9 iterations. Therefore initial feasibility can be maintained using the relaxation technique adopted in RDMPC2 algorithm.

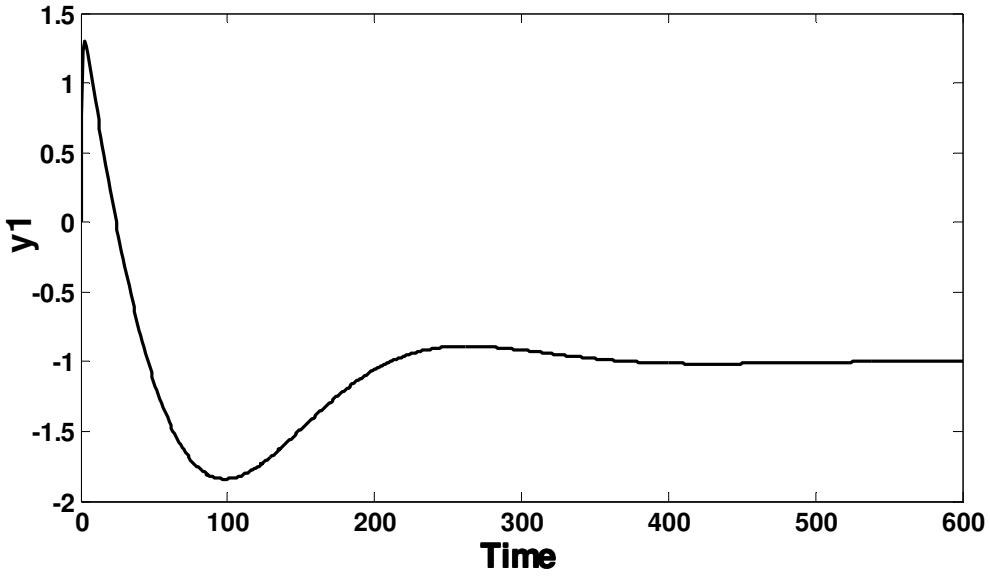


Figure 5.5 Dynamic response of y1 when Nash scheme is used.

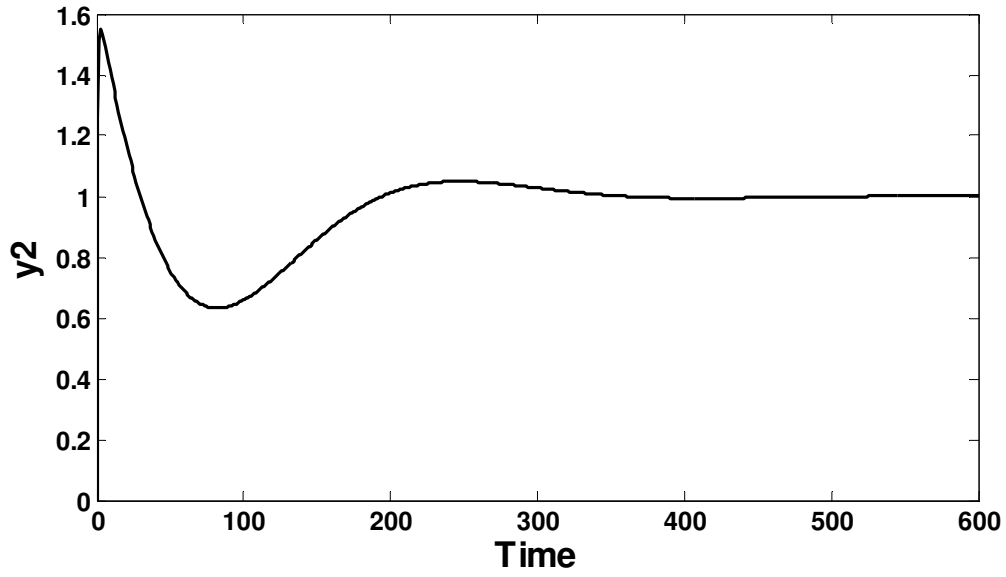


Figure 5.6 Dynamic response of y2 when Nash scheme is used.

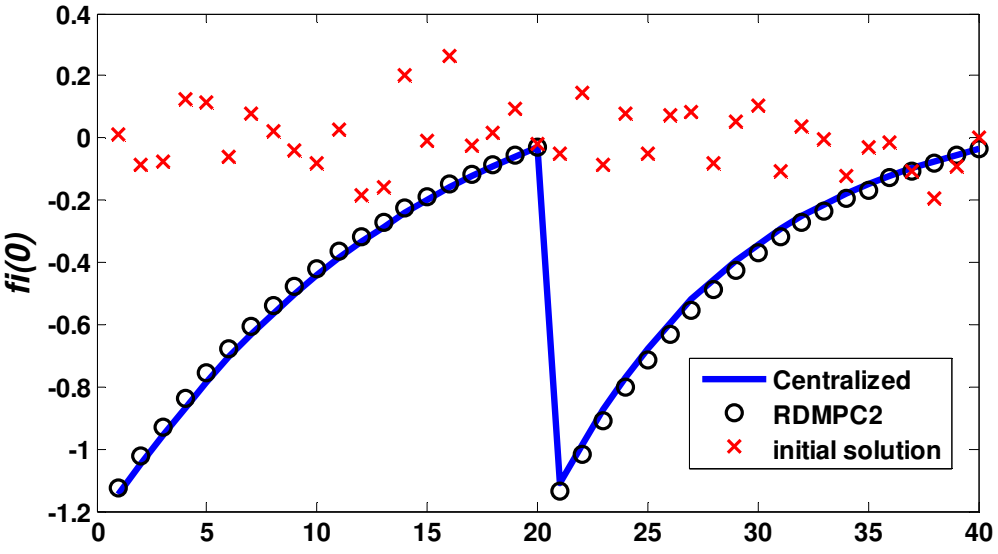


Figure 5.7 Initial feasibility using the relaxation method.

Table (5.1) summarizes the performance comparison between the two algorithms in terms of the resulting sum of squares of errors and corresponding control actions. RDMPC1 cooperative scheme is slightly better than RDMPC2 since the latter was detuned as explained above thus resulting in smaller control actions as can be seen by comparing $\sum_{k=1}^{Ns} \boldsymbol{u}(k)' \boldsymbol{u}(k)$ terms. RDMPC2 Nash scheme performed very poorly by providing a very conservative control action that resulted in a sluggish response. The comparison in terms of the maximum CPU time per control interval indicated that both algorithms required the same time of 4 seconds. It should be remembered that in the RDMPC1 the algorithm was terminated after 10 iterations before convergence. However, as the problem size increases the difference in CPU time between the algorithms becomes significant as shown in the next example.

Table 5.1 Performance comparison between the two algorithms

Algorithm	$\sum_{k=1}^{Ns} \boldsymbol{e}(k)' \boldsymbol{e}(k)$	$\sum_{k=1}^{Ns} \boldsymbol{u}(k)' \boldsymbol{u}(k)$
RDMPC1:		
Cooperative	3.50	6.42
Nash	12.60	1.50
RDMPC2:		
Cooperative	4.05	9.18
Nash	1.11e03	7.28

5.5.2 Example 2

Example 3 from chapter 4 is used. The RDMPC2 algorithm with the cooperative and Nash schemes is applied and compared with RDMPC1. The cooperative scheme parameters are as follows: The unconstrained gains $\boldsymbol{F}_i$ are designed offline using equation (5.11) with $\boldsymbol{S}_1 = \boldsymbol{S}_2 = \boldsymbol{S}_3 = I$, $R_1 = R_2 = R_3 = I$, and the invariant set is maximized using Nc = 12. For the online computations the penalty factor ω = 1e2, and a filter factor for equation (5.16) of α = 0.95 are used. For RDMPC2 the unconstrained gains are designed iteratively using the same weights and same parameters as the ones used for online computations. Figures (5.8) through (5.13) show the dynamic response and control actions when RDMPC2 is implemented for both cooperative and Nash schemes and compared with centralized control. The response is comparable to the one obtained with RDMPC1 as can be shown by comparing with the corresponding figures in the previous chapter. The comparison in terms of performance and maximum CPU time per control interval are summarized in Tables (5.2) and (5.3); respectively.

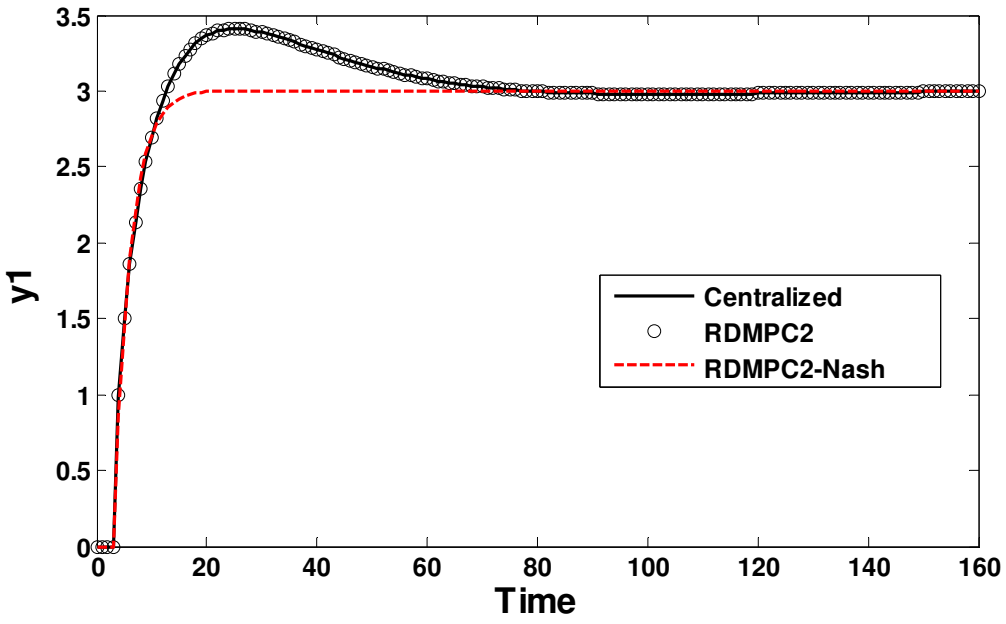


Figure 5.8 Dynamic response of y1 using RDMPC2.

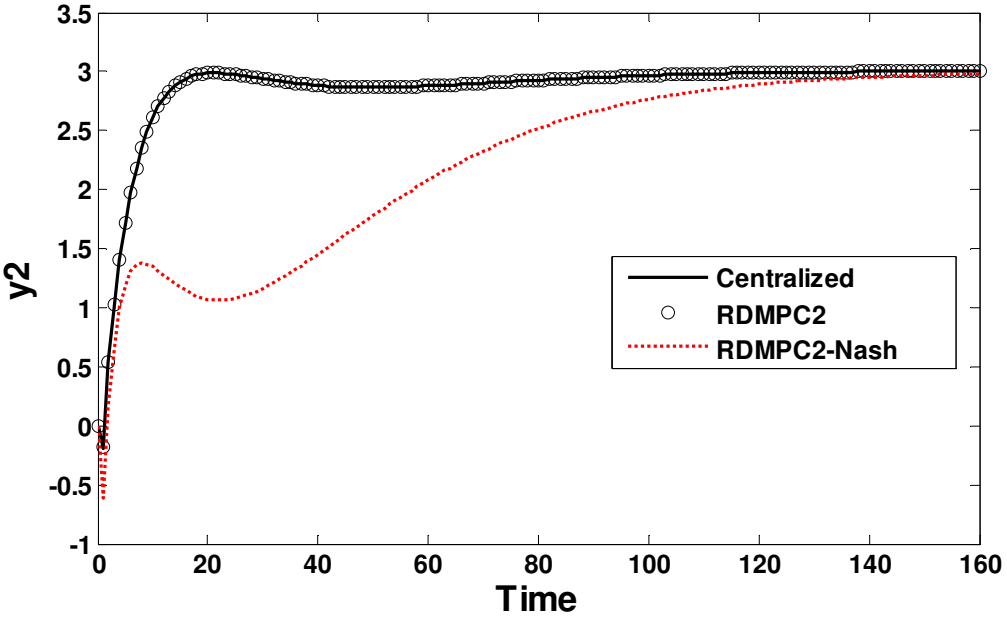


Figure 5.9 Dynamic response of y2 using RDMPC2.

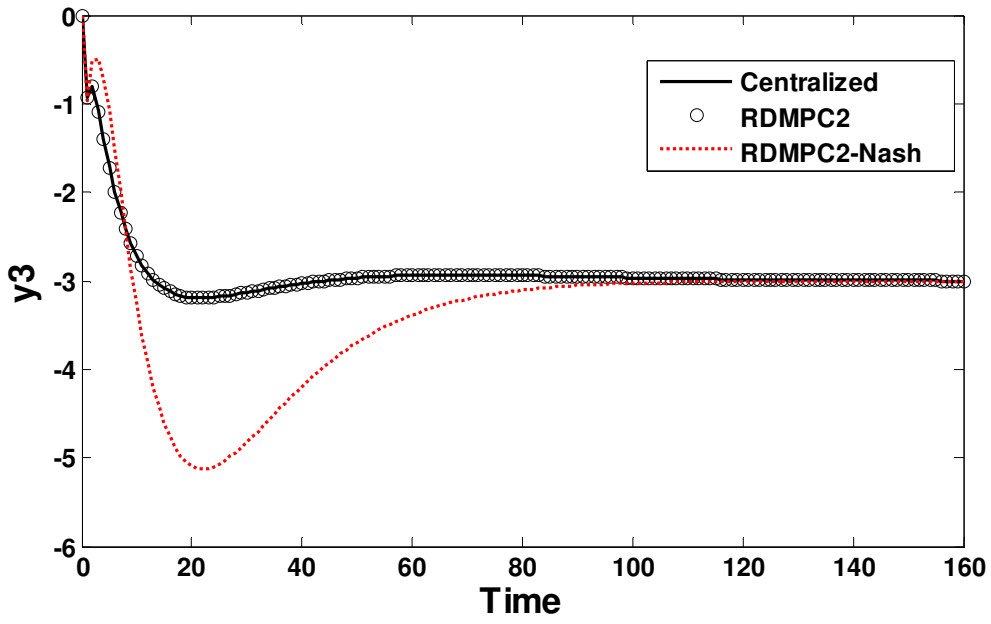


Figure 5.10 Dynamic response of y3 using RDMPC2.

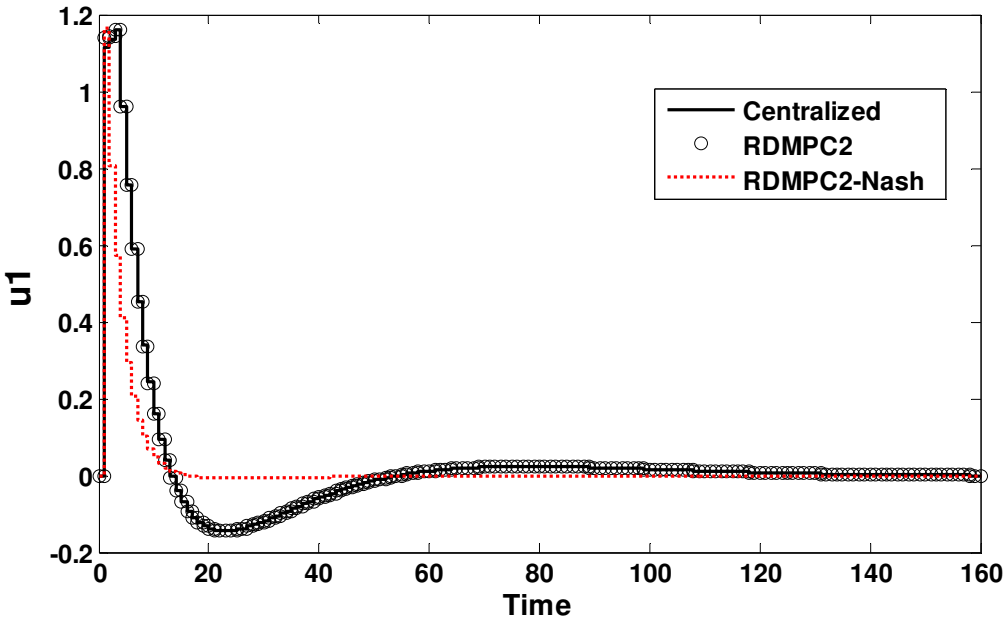


Figure 5.11 Control action u1 using RDMPC2.

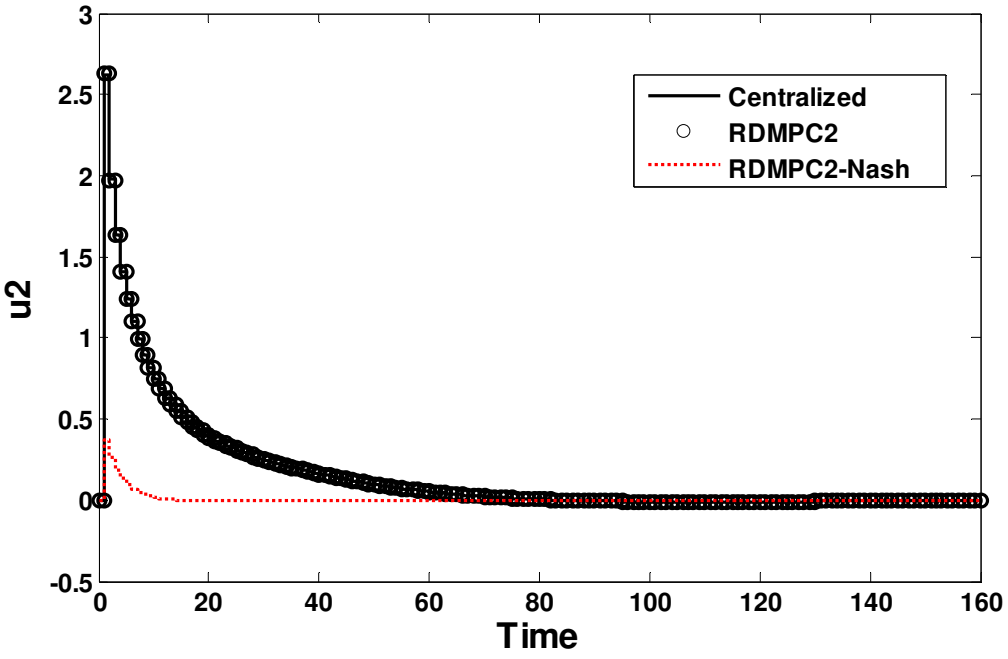


Figure 5.12 Control action u2 using RDMPC2.

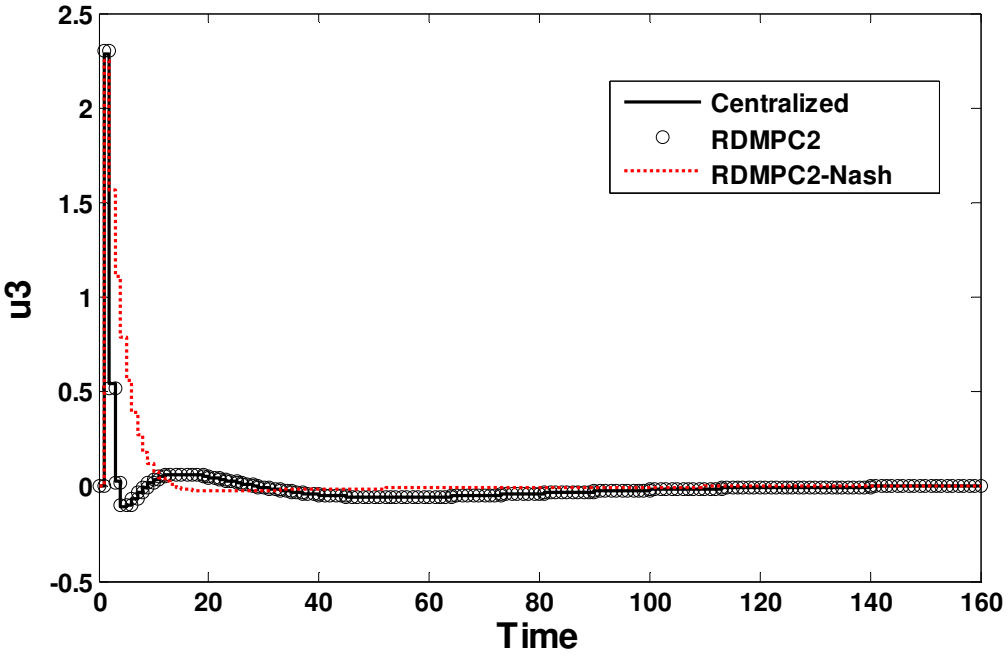


Figure 5.13 Control action u3 using RDMPC2.

Table 5.2 Performance comparison between the two algorithms

Algorithm	$\sum_{k=1}^{Ns} \boldsymbol{e}(k)' \boldsymbol{e}(k)$	$\sum_{k=1}^{Ns} \boldsymbol{u}(k)' \boldsymbol{u}(k)$
RDMPC1:		
Cooperative	121.67	30.21
Nash	376.93	9.16
RDMPC2:		
Cooperative	114.49	37.80
Nash	382.58	12.95

Table 5.3 CPU time per control interval

Algorithm	CPU time, sec
RDMPC1	60
RDMPC2	10

From Table (5.2) RDMPC2 achieved slightly better performance than RDMPC1 but overall both are comparable. In terms of CPU time per control interval RDMPC2 is six times faster than RDMPC1 indicating significant computation efficiency favoring RDMPC2. Therefore it is possible to conclude that for the examples shown above RDMPC2 algorithm can reduce online computations with comparable performance to RDMPC1 while satisfying robust stability and feasibility constraints.

5.6 Conclusions

In this work a new framework for robust DMPC is proposed to reduce online computations while maintaining robust stability and feasibility. The closed-loop dual-mode paradigm was employed in order to perform most of the CPU intensive computations offline using convex optimization to obtain the largest possible invariant sets. The RDMPC2 algorithm requires solving N convex problems in parallel when the cooperative scheme is implemented. On the other hand, it is also possible to use the strategy to satisfy a Nash equilibrium objective function. A relaxation method was incorporated with the algorithm to satisfy initial feasibility by introducing slack variables that converge to zero immediately at early iterations. Two simulation case studies are used to illustrate the algorithm and to compare it with an RDMPC1 algorithm proposed in the previous chapter. It has been shown that the new proposed method (RDMPC2) significantly reduces online computations while providing similar performance as compared to the previous technique (RDMPC1).

CHAPTER 6

Conclusions and Future Remarks

DMPC strategies are used to capitalize on the benefits from using the decentralized structure while achieving improved plant-wide performance and stability via coordination. This work considers the robustness issues related to DMPC strategies in the presence of model uncertainty. The robustness of DMPC with respect to model uncertainty has been identified by researchers as a key factor in the successful application of DMPC. Two main objectives were considered in this work: 1) the development of a systematic methodology for the selection of a DMPC control structure in the presence of model error; 2) the development of novel online algorithms for robust DMPC that explicitly account for model errors. Conclusions drawn from this research are provided below, followed by a summary of future work.

6.1 Conclusions

A new systematic methodology for the selection of a control structure in the context of DMPC was developed. The methodology seeks for a trade-off between performance and simplicity of structure (e.g., a centralized versus decentralized structure) and it is formulated as a multi-objective mixed-integer nonlinear program (MINLP). The multi-objective function is composed of the contribution of two indices: 1) closed-loop performance index computed as an upper bound on the variability of the closed-loop system due to the effect on the output error of either set-point or disturbance input, and 2) a connectivity index used as a measure of the simplicity of the control structure. The

parametric uncertainty in the models is also considered in the methodology through the use of a polytopic model. In polytopic representations the actual process is assumed to behave inside a polytope whose vertices are defined by linear models that can be obtained from either linearizing a nonlinear model or identification around different operating conditions. The system closed-loop performance and stability are formulated as LMI problems so that the efficient interior-point methods can be exploited. To solve the MINLP a multi-start approach is adopted in which many starting points are generated in an attempt to obtain better solutions close to global optima. The efficiency of the proposed methodology is shown through its application to benchmark simulation examples. The simulation results are consistent with the conclusions obtained from the analysis. The proposed methodology can be applied at the design stage to select the best control configuration in the presence of model errors. The analysis results were found to be somewhat conservative since the performance index is computed based on a worst case scenario that can be difficult to obtain using simulation. Furthermore, the proposed method did not consider constraints.

In chapter 4, a novel algorithm for robust DMPC was developed that explicitly accounts for parametric uncertainty in the model. The algorithm requires the decomposition of the entire system into N subsystems and the solution of N convex optimization problems in parallel in order to minimize an upper bound on a robust performance objective by using a time-varying state-feedback controller for each subsystem. Model uncertainty is explicitly considered through the use of a polytopic model. The algorithm employs a method that has been proven efficient: the LMI

approach, in which the solutions are convex and obtained in polynomial time. An observer is designed and embedded within each controller to perform state estimations and the stability of the observer is checked online via LMI conditions. An iterative design method is also proposed for computing the observer gain. Due to the use of LMI's the algorithm is fast making it attractive for real time implementationIt has been shown that upon convergence the proposed algorithm can achieve the theoretical performance of centralized control. Furthermore, the proposed algorithm can be formulated using a variety of objectives, such as Nash equilibrium, that suits the situation when the interacting processing units are operated by independent agents each trying to satisfy its own independent optimal objective, and for fully decentralized control in the case of communication failure. Such cases are commonly encountered in the process industry. Simulations examples are considered to illustrate the application of the proposed method. However, it was found that as the problem size increases the iterative nature of the proposed scheme becomes computationally demanding which required the need for more efficient strategies as proposed in Chapter 5. Furthermore, the main assumption in this thesis is that there is a reliable communication network thus communication failures and delays were not considered. One possibility is to use the decentralized structure once the failure takes place and switch back to the original scheme when the communication is established again. Potential solutions are discussed in the future remarks section.

In Chapter 5, a new algorithm was developed to improve the online computational efficiency. A dual-mode controller was employed in order to perform most of the heavy computations offline using convex optimization to obtain the largest possible invariant

sets thus rendering the following iterative online solution less conservative. The solution requires satisfying relatively simple constraints and the solution of subproblems each with a small number of decision variables. The algorithm requires solving *N* convex LMI problems in parallel when a cooperative scheme was chosen. The option of using Nash scheme formulation is also available. A relaxation method was incorporated within the algorithm to satisfy initial feasibility by introducing slack variables that converge to zero after a few iterations. Simulated case studies have illustrated the applicability of this approach and have demonstrated that significant improvement can be achieved with respect to computation times as compared to the online method proposed in Chapter 4. However, a possible limitation to this algorithm as compared to the one in Chapter 4 is that although the offline computations of invariant sets reduce the computation time there is no systematic way to choose the length of the control horizon *Nc* to ensure initial feasibility other than via several simulations. Consequently, detuning of the controller may become necessary to avoid using large control horizons as it became obvious in the case of the Nash-equilibrium based scheme.

6.2 Future Remarks

In this section a summary of future work some of which can be seen as extension to the current work is provided in the next paragraphs.

The existing DMPC algorithms are still lacking capabilities such as robustness with respect to actuator failures and/or measurement loss. Also, the performance of the current strategies depends on reliable communication networks. Developing new alternative stand-by algorithms that can be used when such failures occur with a

framework that switches back to the normal operation is of great importance. The recent developments in fault-tolerant control can be extended to DMPC strategies to cope up with failures (Mhaskar 2006; Gandi and Mhaskar 2009). Regarding communication failures or delay, the solutions that have to be exchanged via communication can be approximated with either simple linear models or, if necessary, nonlinear predictors such as artificial neural networks. These nonlinear models could then be embedded within each controller and triggered whenever the corresponding controller losses contact with some of or all the other controllers to provide estimates for their actions. The selection of training and validating data will become difficult with the problem size and number of subsystems.

The current DMPC methods employ conventional optimization algorithms originally developed for centralized MPC (Scattolini 2009). Consequently, developing new algorithms tailored specifically for DMPC can improve the online computations and thus widen the spectrum of applications where this technology can be applied. Recent developments in online optimization can be utilized to formulate new customized algorithms to exploit the structure in DMPC problems such as the approach proposed recently by Wang and Boyd (2008). In the current work a methodology for the selection of control structure for DMPC was proposed. However, the interaction between process design and control was not considered in this methodology. In the context of simultaneous design and control, the centralized MPC strategies have been considered by solving as a single-level mathematical program with complementarily constraints (Baker and Swartz (2008). This approach may be extended to DMPC strategies to seek for optimal integration of control and design of process systems. By the proper choice of

a cost term, to be incorporated in an economic cost function that penalizes the complexity of DMPC structure and by introducing binary variables that change the DMPC structure between fully decentralized to fully connected, the resulting mixed-integer nonlinear programming problem may find an optimal trade-off between structure simplicity, closed-loop robust performance and a plant design economic optimum. This problem can also be readily expanded to consider reconfigurable hierarchical structures to cope with time-varying performance and constraints that are widely encountered in the control and operation of complex industrial systems as occurring during start-ups and shut-downs of industrial processes.

To achieve optimal economic operation and plant-wide control, the integration of different layers of the process automation hierarchy, e.g., supply-chain/planning, real time optimization (RTO), two-stage MPC, and plant layers, is of paramount importance (Tatjewski 2008; Scattolini 2009). It has been reported that poor performance in the two-stage MPC layer is not uncommon due to the effects of feedback from the plant layer (Nikandrov and Swartz 2009). A challenging problem, therefore, is to improve the performance of the overall system through the use of designs that account for the uncertainties in models and demands and that can achieve integration across the layers mentioned above. The results to be obtained from this research are expected to have a significant impact on the process industry by improving economic performance under conditions that include economic and model uncertainties.

REFERENCES

Al-Gherwi, W., Budman, H., and Elkamel, A., (2008) "Robustness issues related to the application of distributed model predictive control strategies", *proceedings of the 17th World Congress IFAC*, Seoul, Korea, 8395-8400.

Al-Gherwi, W., Budman, H., and Elkamel, A. (2009) "An online algorithm for robust distributed Model Predictive Control," *Proceedings of the Advanced Control of Chemical Processes* ADCHEM, Paper no. 33, Turkey.

Al-Gherwi, W., Budman, H., and Elkamel, A. (2010) "Selection of control structure for distributed model predictive control in the presence of model errors" *Journal of Process Control*, **20**, 270-284.

Aplevich, J. D. (2000) "The essentials of linear state-space systems", John Wiley & sons, USA.

Arkun, Y. (1986) "Dynamic block relative gain and its connection with the performance and stability of decentralized control structures", *International Journal of Control*, **46**, 1187-1196.

Baker, R., and Swartz, C. L. E. (2008) Interior point solution of multilevel quadratic programming problems on constrained model predictive control applications. *Ind. Eng. Chem. Res.*, **47**, 81 – 91.

Bars, R., Colaneri, P., de Souza, C.E., Dugard, L., Allgower, F., Kleimenov, A., and Scherer, C. (2006) "Theory, algorithms, and technology in the design of control systems", Control Engineering Practice, **30**, 19-30.

Bemporad, A., Morari, M., Dua, V., and Pistikopoulos, N. (2002) "The explicit linear quadratic regulator for constrained systems", *Automatica*, **38**, 3-20.

Braatz, R. D., Lee, J. H., and Morari, M. (1996) "Screening plant designs and control structures for uncertain systems", *Computers and Chemical Engineering*, **20** (4), 463-468.

Bristol, E. H. (1966) "Measure of interactions for multivariable process control" *IEEE Trans. Automatic Control*, AC-11, 133-134.

Boyd, S., El.Ghaoui, L., Feron, E., and Balakrishnan, V. (1994). "Linear matrix inequalities in system and control theory", SIAM, USA.

Camacho, E. F., and Bordons, C. (2003) "Model predictive control", 2nd edition, Springer-Verlag, London, UK.

Camponogara, E., Jia, Dong, Krough, Bruce H., and Talukdar, S. (2002) "Distributed model predictive control" *IEEE Control Systems Magazine*, 44-52.

Charos G.N., and Arkun, Y. (1993) "A decentralized quadratic dynamic matrix control algorithm", *Journal of Process Control*, **3**, 75-83.

Cheng, R., Forbes, J.F., and Yip, W.S. (2007) “Price-driven coordination method for solving plant-wide MPC problems”. *J. Proc. Cont.*, **17**, 429.

Cheng, R., Forbes, J.F., and Yip, W.S. (2008) “Dantzig-Wolfe decomposition and plant-wide MPC coordination". *Comput. Chem. Eng.*, **32**, 1507.

Chen, Dan, and Seborg, Dale E. (2002) “Relative gain array analysis for uncertain process models”, *AIChE*, **48** (2), 302-310.

Chu, D. (2006) “Explicit robust model predictive control and its applications”, PhD thesis, University of Alberta, Edmonton, Canada.

Cutler, C. R., Morshedi, A., and Haydel, J. (1983) “An industrial perspective on advanced control”, *AIChE annual meeting*, Washinton, DC.

Cutler, C. R., and Ramaker, B. L. (1979) “Dynamic matrix control a computer control algorithm”, *AIChE annual meeting*, Houston, TX.

Doyle, J. C., and Packard, A. (1987) “Uncertain multivariable systems from a state space perspective”, *ACC proceedings*, New Jersey, 2147-2152.

Du, Xiaoning, Xi, Yugeng, and Li, Shaoyuan (2001) “Distributed model predictive control for large-scale systems”, *Proceedings of the American Control Conference*, 3142-3143.

Dunbar, W. B. (2005) "A distributed receding horizon control algorithm for dynamically coupled nonlinear systems", *Proceedings of the 44th IEEE CDC and European Control Conference*, 6673-6679.

Gandhi, R. and P. Mhaskar (2009) "A safe-parking framework for plant-wide fault-tolerant control", *Chemical Engineering Scince.,* **64**, 3060-3071.

Gao, J. (2004) "Robust control design of gain-scheduled controllers for nonlinear processes", PhD thesis, University of Waterloo.

Gao, J. and Budman, H. (2005) "Design of robust gain-scheduled PI controllers for nonlinear processes", *Journal of process control,* **15,** 807-817.

Garcia, C. E., and Morshedi, A. M. (1986) "Quadratic programming solution of dynamic matrix control (QDMC)", *Chemical Engineering Communications*, **46**, 73-87.

Grosdidier. P., and Morari, M. (1986) "Interaction measures for systems under decentralized control", *Automatica*, **22** (3), 309-319.

Grosdidier. P., and Morari, M. (1987) "The μ interaction measure", *Ind. Eng. Chem. Res.*, **26,** 1193-1202.

Jia, Dong, and Krogh, Bruce (2001) "Distributed model predictive control", *Proceedings of the American Control Conference*, 2767-2772.

Jia, Dong, and Krogh, Bruce (2001) "Min-max feedback model predictive control for distributed control with communication", *Proceedings of the American Control Conference*, 4507-4512.

Kalman, (1960) "A new approach to linear filtering and prediction problems", *Journal of Basic Engineering*, **87**, 35-45.

Kariwala, V., Forbes, J.F., and Meadows, E.S., "Block relative gain: properties and pairing rules", *Industrial and Engineering Chemistry Research*, **42**, 4564–4574.

Katebi, M. R., and Johnson, M. A., (1997) "Predictive control design for large-scale systems", *Automatica*, **33** (3), 421-425.

Kothare, M., Balakrishnan, V., and Morari, M. (1996) "Robust constrained model predictive control using linear matrix inequalities", *Automatica*, **22** (10), 1361-1379.

Kouvaritakis, B., Rossiter, J.A., and Schurmans (2000) "Effiecient robust predictive control", IEEE Transactions on Automatic Control, 45, 1545-1549.

Li, Shaoyuan, Zhang, Yan, and Zhu, Quanmin (2005) "Nash-optimization enhanced distributed model predictive control applied to shell benchmark problem", *Information Sciences*, **170**, 329-349.

Lee, P. L., Li, H., and Cameron, I. T., (2000) "Decentralized control design for nonlinear multi-unit plants: a gap metric approach", *Chemical Engineering Science*, **55** 3743-3758.

Li, S., Lim, K., and Fisher, (1989) “A state space formulation for model predictive control”, *AIChE*, **35**, 241-249.

Lunstrom, P., Lee, J., and Morari, M. (1995) “Limitations of dynamic matrix control”, *Computers and Chemical Engineering*, **19** (4), 409-421.

Lunze, Jan (1992) “Feedback control of large-scale systems”, Prentice Hall, London, UK.

Lee, P. L., Li, Huaizhong, and Cameron, I. T. (2000) “Decentralized control design for nonlinear multi-unit plants: a gap metric approach”, *Chemical Engineering Science*, **55**, 3743-3758.

Li, S., Zhang, Y., and Zhu, Q. (2005) “Nash-optimization enhanced distributed model predictive control applied to the shell benchmark problem”, *information science*, **170**, 329-349.

Liu, J., de la Pena, D.M., and Christofides, P.D. (2009) “Distributed model predictive control of nonlinear process systems”, *AIChE*, **55**, 1171.

Löfberg, J. (2004) “YALMIP : A toolbox for modeling and optimization in MATLAB”, *In Proceedings of the CACSD Conference*, Taipei, Taiwan.

Maciejowski, J. M. (2002) “Predictive control with constraints”, Prentice Hall, London, UK.

Manousiouthakis, V., Savage, R., and Arkun, Y. (1986) "Synthesis of decentralized process control structures using the concept of block relative gain", *AIChE*, **32** (6), 991-1003.

Magni, L., and Scattolini, R. (2006) "Stabilizing decentralized model predictive control of nonlinear systems", *Automatica*, **42**, 1231-1236.

Marcos, N. I., Forbes, J.F., and Guay, M. (2009) "Coordination of distributed model predictive controllers for constrained dynamic processes", *ADCHEM*, Turkey.

Morari, M., and Zafiriou, E. (1989) "Robust process control", Prentice Hall, NJ, USA.

Mercangöz, M. and Doyle III, F. J. (2007) "Distributed model predictive control of an experiental four-tank system", *Journal of Process Control*, **17**, 297-308.

Mhaskar, P. (2006) "Robust model predictive control design for fault-tolerant control of process systems", *Industrial & Engineering chemistry Research*, **45**, 8565-8574.

Motee, Nader, and Sayyar-Rodsari, Bijan (2003) "Optimal partitioning in distributed model predictive control", *Proceedings of the American control Conference*, 5300-5305.

Muske, K. B., and Rawlings, J. B. (1993) "Model predictive control with linear models", *AIChE*, **39** (2), 262-287.

Negenborn, R. R., De Schutter, B., and Hellendoorn, J. (2004) "Multi-agent model predictive control: A survey", Technical report 04-010, Delft University of Technology, The Netherlands.

Nikandrov, A., and Swartz, C. L. E. (2009) Sensitivity analysis of LP-MPC cascade control systems. *J. Process Control*, **19**, 16-24.

Pistikopoulos, E., Georgiadis, M, and Dua, V., (2007) "Multi-Parametric model-based control", Wiley-VCH.

Prett, D. M., and Garcia, C. E. (1988) "Fundamental process control", Butterworths, USA.

Qin, S. J., and Badgwell, T. A. (2003) "A survey of industrial model predictive control technology", *Control Engineering Practice*, **11**, 733-764.

Rawlings, J. B. (2000) "Tutorial overview of model predictive control", *IEEE Control Systems Magazine*, 38-52.

Rawlings, James B., and Stewart, Brett T. (2008) "Coordinating multiple optimization-based controllers: new opportunities and challenges", *Journal of Process Control*, **18**, 839-845.

Richalet, J. Rault, A., Testud, J. L., and Papon, J. (1978) "Model predictive heuristic control: applications to industrial processes", *Automatica*, **14**, 413-428.

Samyudia, Y., Lee, P. L., and Cameron, I. T. (1994) "A methodology for multi-unit control design", *Chemical Engineering Science*, **49** (23), 3871-3882.

Samyudia, Y., Lee, P. L., and Cameron, I. T. (1995) "A new approach to decentralized control design", *Chemical Engineering Science*, **50** (11), 1695-1706.

Scattolini, R., (2009) "Architecture of distributed and hierarchical model predictive control – A review", *Journal of Process Control*, **19**, 723-731.

Seborg, Dale E., Edgar, Thomas F., and Mellichamp, Duncan A. (2004) "Process dynamics and control", John Wiley, USA.

Sceiber, S. (2004) "Decentralized control", *Control Engineering*, **51** (4), 44.

Skogestad, Sigurd, and Morari, M. (1988) "Robust control of ill-conditioned plants: high-purity distillation", *IEEE Transactions on Automatic Control*, **12**, 1092-1104.

Skogestad, S. (2000) "Plantwide control-A review and a new design procedure", *Modeling, Identification and Control*, **21** (4), 209-240.

Skogestad, S. (2004) "Control structure design for complete chemical plants", *Computers and Chemical Engineering*, **28**, 219-234.

Skogestad, S., and Postlethwaite, I. (2005) "Multivariable feedback control analysis and design", John Wiley, England.

Sun, Y., and El-Farra, N. (2008) "Quasi-decentralized model-based networked control of process systems", *Computers and Chemical Engineering*, **32**, 2016-2029.

Tatjewski, P. (2008) Advanced control and on-line process optimization in multilayer structures. *Annual Reviews in Control*, **32**, 71-85.

Vaccarini, M., Longhi, S., and Katebi, M.R. (2009) "Unconstrained networked decentralized model predictive control", *Journal of Process Control*, **19**, 328-339.

Vadigepalli, R., and Doyle III, F. J. (2003) "Structural analysis of large-scale systems for distributed state estimation and control applications", *Control Engineering Practice*, **11** (8), 895-905.

VanAntwerp, J.G. and Braatz, R.D. (2000) "A tutorial on linear and bilinear matrix inequalities", *Journal of Process Control*, **10**, 363-385.

Venkat, Aswin N. (2006) "Distributed model predictive control: theory and applications", PhD thesis, University of Wisconsin-Madison.

Wan, Z., and Kothare, M. V. (2003) "An efficient off-line formulation of robust model predictive control using linear matrix inequalities", *Automatica*, **39**, 837-846.

Wang, D.H., and Soh, C.B. (2000) "Adaptive neural model-based decentralized predictive control", *International Journal of Systems Science*, **31**, 119-130.

Wang, Y., and Boyd, S. (2008) Fast model predictive control using online optimization. *Proceedings of the 17th World Congress, IFAC*, Seoul, Korea, 6974-6979.

Xu, S., and Bao, J. (2009) "Distributed control of plantwide chemical processes", *Journal of Process Control*, **19**, 1671-1687.

Xu, X. M., Xi, Y. G., and Zhang, Z. J. (1988) "Decentralized model predictive control (DCP) of large scale systems", *Information and Decision Technologies*, **14** (3), 307-322.

Zhang, Y., and Li, S. (2007) "Netwroked model predictive control based on neighbourhood optimization for serially connected large-scale processes", *Journal of Process Control*, **17**, 37-50.

Zhu, G-Y, Henson, M., and Ogunnaike, B. (2000) "A hybrid model predictive control strategy for nonlinear plant-wide control", *Journal of Process Control*, **10**, 449-458.

Zhu, G-Y, and Henson, M. (2000) "Model predictive control of interconnected linear and nonlinear process", *Industrial Engineering Chemistry Research*, **41**, 801-816.

APPENDIX: Basic MATLAB Codes

Key MATLAB codes used in Chapter 3

```
function gamma=Glmidmpc(lambda)

if any(lambda)<0|any(lambda)>1

    gamma=1e6;
    return;
end


alfa=0.99;
Ts=1;
dt1=lambda(3);
dt2=lambda(4);
% dt1=1;
% dt2=1;
A1=[-1/75 0;0 -1/75];B1=[.878/75 .864/75;1.082/75 1.096/75];C1=eye(2);D1=zeros(2);
Plant1=ss(A1,B1,C1,D1);
[Ap{1},Bp{1},Cp{1},Dp{1}]=ssdata(c2d(Plant1,Ts));

A2=[-1/75 0;0 -1/75];B2=[.878*1.8/75 .864*.2/75;1.082*1.8/75
1.096*.2/75];C2=eye(2);D2=zeros(2);
Plant2=ss(A2,B2,C2,D2);
[Ap{2},Bp{2},Cp{2},Dp{2}]=ssdata(c2d(Plant2,Ts));


A1=[-1/75 0;0 -1/75];B1=[.878/75 .864*dt1/75;1.082/75*dt2
1.096/75];C1=eye(2);D1=zeros(2);
Plant1=ss(A1,B1,C1,D1);
[A,B,C,D]=ssdata(c2d(Plant1,Ts));


Interval=2;

%[Am,Bm,Cm,Dm]=ssdata(Plant_ss); %state-space matrices
%models for subsystems
%subsystem 1
A1=A(1,1);A2=A(2,2);
B11=B(1,1);B12=B(1,2);B21=B(2,1);B22=B(2,2);
C1=C(1,1);C2=C(2,2);
ywt=[1 1];
Gs=[];
Glmi=[];

uwt=[lambda(1) lambda(2)];
%uwt=[];
ywt1=1;ywt2=1;
%uwt1=1.4203e-003;uwt2=3.1250e-006;
uwt1=uwt(1);uwt2=uwt(2);
p1=20;p2=20;m1=5;m2=5;

nu1=1;ny1=1;nu2=1;ny2=1;
%create the input and output weighting matrices
wu1=[];
wu2=[];
wy1=[];
wy2=[];
for i=1:m1*ny1
    wu1=[wu1 uwt1];
end
for i=1:m2*ny2
    wu2=[wu2 uwt2];
end
```

```
for i=1:p1*ny1
    wy1=[wy1 ywt1];
end
for i=1:p2*ny2
    wy2=[wy2 ywt2];
end
Wu1=diag(wu1);
Wu2=diag(wu2);
Wy1=diag(wy1);
Wy2=diag(wy2);

EY01=eye(ny1);EY1=EY01;
for i=1:p1-1
    EY1=[EY1 EY01];
end

N1=EY1';

EY02=eye(ny2);EY2=EY02;
for i=1:p2-1
    EY2=[EY2 EY02];
end

N2=EY2';

%create the controller's paramater and Kmpcs
%controller1
[nx1,nx1]=size(A1);
[nx2,nx2]=size(A2);
Epsi1=zeros(ny1*p1,nx1);

for i=1:p1
    Epsi1((i-1)*ny1+1:i*ny1,:)=C1*A1^i;
end

%Epsi12=zeros(ny1*p1,nx2);

%for i=1:p1
 %   Epsi12((i-1)*ny1+1:i*ny1,:)=C12*A2^i;
%end

Temp1=C1*B11;
GA1=Temp1;
for i=1:p1-1
    for j=1:i
        Temp1=Temp1+C1*A1^j*B11;
    end
    GA1=[GA1;Temp1];
    Temp1=C1*B11;
end

Temp12=C1*B12;
GA12=Temp12;
for i=1:p1-1
    for j=1:i
        Temp12=Temp12+C1*A1^j*B12;
    end
    GA12=[GA12;Temp12];
    Temp12=C1*B12;
end

Theta1 = zeros(ny1*p1,nu1*m1);
Theta1(1:p1*ny1,1:nu1)=GA1;

for i =2:m1
    Theta1((i-1)*ny1+1:p1*ny1,(i-1)*nu1+1:i*nu1)=Theta1(1:(p1-(i-1))*ny1,1:nu1);
end
Theta12 = zeros(ny1*p1,nu2*m1);
Theta12(1:p1*ny1,1:nu2)=GA12;

for i =2:m1
```

```
    Theta12((i-1)*ny1+1:p1*ny1,(i-1)*nu2+1:i*nu2)=Theta12(1:(p1-(i-1))*ny1,1:nu2);
end

Kfull1=inv(Theta1'*Wy1'*Wy1*Theta1+Wu1'*Wu1)*Theta1'*Wy1'*Wy1;
Kmpc1=Kfull1(1:nu1,:);

%create the controller's paramater and Kmpcs
%Controller 2
[nx2,nx2]=size(A2);

Epsi2=zeros(ny2*p2,nx2);

for i=1:p2
    Epsi2((i-1)*ny2+1:i*ny2,:)=C2*A2^i;
end

%Epsi21=zeros(ny2*p2,nx1);

%for i=1:p2
%    Epsi21((i-1)*ny2+1:i*ny2,:)=C21*A1^i;
%end

Temp2=C2*B22;
GA2=Temp2;
for i=1:p2-1
    for j=1:i
        Temp2=Temp2+C2*A2^j*B22;
    end
    GA2=[GA2;Temp2];
    Temp2=C2*B22;
end

Temp21=C2*B21;
GA21=Temp21;
for i=1:p2-1
    for j=1:i
        Temp21=Temp21+C2*A2^j*B21;
    end
    GA21=[GA21;Temp21];
    Temp21=C2*B21;
end

Theta2 = zeros(ny2*p2,nu2*m2);
Theta2(1:p2*ny2,1:nu2)=GA2;

for i =2:m2
    Theta2((i-1)*ny2+1:p2*ny2,(i-1)*nu2+1:i*nu2)=Theta2(1:(p2-(i-1))*ny2,1:nu2);
end

Theta21 = zeros(ny2*p2,nu1*m2);
Theta21(1:p2*ny2,1:nu2)=GA21;

for i =2:m2
    Theta21((i-1)*ny2+1:p2*ny2,(i-1)*nu1+1:i*nu1)=Theta21(1:(p2-(i-1))*ny2,1:nu1);
end

Kfull2=inv(Theta2'*Wy2'*Wy2*Theta2+Wu2'*Wu2)*Theta2'*Wy2'*Wy2;
Kmpc2=Kfull2(1:nu2,:);
D1=[Kfull1 0*Kfull1;0*Kfull2 Kfull2];
D0=[0*Kfull1*Theta12 -Kfull1*Theta12;-Kfull2*Theta21 0*Kfull2*Theta21];

K=inv(eye(size(D0))-D0)*D1;
Kmpc=[K(1,:);K(m1+1,:)];
Epsi=[Epsi1 0*Epsi1;0*Epsi2 Epsi2];GA=[GA1 GA12;GA21 GA2];

nu=2;ny=2;p=p1;m=m1;nx=nx1+nx2;
Matrix1=[1 0];
for i=1:p-1
    Matrix1=[Matrix1;[1 0]];
end
Matrix2=[0 1];
```

```
for i=1:p-1
    Matrix2=[Matrix2;[0 1]];
end
Matrix=[Matrix1;Matrix2];
N2=Matrix;
for i=1:Interval

%Xk+1
A11=Ap{i}-Bp{i}*Kmpc*N2*Cp{i};
A12=Bp{i}*Kmpc*(N2*C-Epsi);
A13=Bp{i}-Bp{i}*Kmpc*GA;
A14=Bp{i}*Kmpc*Matrix;


%Xck+1
A21=-B*Kmpc*N2*Cp{i};
A22=A+B*Kmpc*(N2*C-Epsi);
A23=B-B*Kmpc*GA;
A24=B*Kmpc*Matrix;

%Uk
A31=-Kmpc*N2*Cp{i};
A32=Kmpc*(N2*C-Epsi);
A33=eye(nu,nu)-Kmpc*GA;
A34=Kmpc*Matrix;

%Rk+1
A41=zeros(ny,nx);
A42=zeros(ny,nx);
A43=zeros(ny,nu);
A44=(alfa)*[1 0;0 0];
%aa=[A11 A12 A13;A21 A22 A23;A31 A32 A33];
%AA=[A11 zeros(2,2) A12 A13;zeros(2,2) A11 A12 A13;.5*A21 .5*A21 A22 A23;A31 A31 A32
A33];

Asys{i}=[A11 A12 A13 A14;A21 A22 A23 A24;A31 A32 A33 A34;A41 A42 A43 A44];
%Asys{i}=[A11 A12 A13;A21 A22 A23;A31 A32 A33];

%Asys=AA;
B1=zeros(nx,ny);
B2=zeros(nx,ny);
B3=zeros(nu,ny);
B4=(1-alfa)*[1 0;0 0];
%bb=[B1;B2;B3];
Bsys=[B1;B2;B3;B4];

%Csys=[-Cp*(Ap-Bp*Kmpc*N2*Cp),-Cp*(Bp*Kmpc*(N2*C-Epsi)),-Cp*(Bp-
Bp*Kmpc*GA),alfa*eye(ny,ny)-Cp*Bp*Kmpc*alfa*Matrix];
%Csys=[-Cp*(Ap-Bp*(Kmpc*N2*Cp)),-Cp*(Bp*Kmpc*(N2*C-Epsi)),-Cp*(Bp-Bp*Kmpc*GA)];

Csys{i}=[-Cp{i},zeros(ny,nx),zeros(ny,nu),eye(ny,ny)];
%Dsys=(1-alfa)*eye(ny,ny)-Cp*Bp*Kmpc*(1-alfa)*Matrix;
%Dsys=(1-alfa)*eye(ny,ny);
Dsys{i}=[0 0;0 0];%zeros(ny,ny);
MPCsys{i}=ss(Asys{i},Bsys,Csys{i},Dsys{i},Ts);
%Poles=pole(MPCsys);

end

A1=Asys{1};
A2=Asys{2};
B1=Bsys;
B2=Bsys;
C1=Csys{1};
C2=Csys{2};
D1=Dsys{1};
D2=Dsys{2};


[ns,ns]=size(Asys{1});
```

```
setlmis([]);
Po=lmivar(1,[ns 1]);%Po is symmetric block diagonal
Gamm=lmivar(1,[1 1]);%

%LMI#1
lmiterm([1 1 1 Po],A1',A1);%A'*Po*A
lmiterm([1 1 1 Po],-1,1);%-Po
lmiterm([1 1 2 Po],A1',B1);%A'*Po*B
lmiterm([1 1 3 0],C1');%C'
lmiterm([1 2 2 Po],B1',B1);%B'*Po*B
lmiterm([1 2 2 Gamm],-1,1);%-gamma2I
lmiterm([1 2 3 0],D1');%D'
lmiterm([1 3 3 0],-1);%1

%LMI#2
lmiterm([2 1 1 Po],A2',A2);%A'*Po*A
lmiterm([2 1 1 Po],-1,1);%-Po
lmiterm([2 1 2 Po],A2',B2);%A'*Po*B
lmiterm([2 1 3 0],C2');%C'
lmiterm([2 2 2 Po],B2',B2);%B'*Po*B
lmiterm([2 2 2 Gamm],-1,1);%-gamma2I
lmiterm([2 2 3 0],D2');%D'
lmiterm([2 3 3 0],-1);%1

%LMI#3
lmiterm([3 1 1 Po],-1,1);
try


LMIsys=getlmis;
CC=mat2dec(LMIsys,zeros(ns),1);
[gam,xopt]=mincx(LMIsys,CC,[1.0*exp(-6) 100 1e9 500 1]);
Qm = dec2mat(LMIsys,xopt,Gamm);
catch
  gam=100;
end

%P = sdpvar(ns,ns);
%t = sdpvar(1);
%objective = t;
%L1=[A1'*P*A1-P A1'*P*B1 C1';B1'*P*A1 B1'*P*B1-t*eye(size(2)) D1';C1 D1 -eye(2)];
%L2=[A2'*P*A2-P A2'*P*B2 C2';B2'*P*A2 B2'*P*B2-t*eye(size(2)) D2';C2 D2 -eye(2)];
%F = set(P>0) + set(L1<0)+ set(L2<0);
%options = sdpsettings('solver','sedumi','verbose',0);
%options = sdpsettings('solver','sedumi');
%solvesdp(F,objective,options);
%double(P);
%gammasedume = sqrt(double(t))


gamma=sqrt(gam);

if dt1==0
    Index1=0;
else
    Index1=1;
end
if dt2==0
    Index2=0;
else
    Index2=1;
end
Index=(Index1+Index2)/2.0;

gamma = 1*gamma + 0.*Index;


alfa=0.99;
Ts=1;
dt=1;
A1=[-1/75 0;0 -1/75];B1=[.878/75 .864/75;1.082/75 1.096/75];C1=eye(2);D1=zeros(2);
```

```
Plant1=ss(A1,B1,C1,D1);
[Ap{1},Bp{1},Cp{1},Dp{1}]=ssdata(c2d(Plant1,Ts));

A2=[-1/75 0;0 -1/75];B2=[.878*1./75 .864*1/75;1.082*1./75
1.096*1/75];C2=eye(2);D2=zeros(2);
Plant2=ss(A2,B2,C2,D2);
[Ap{2},Bp{2},Cp{2},Dp{2}]=ssdata(c2d(Plant2,Ts));

A=Ap{1};B=Bp{1};C=Cp{1};D=Dp{1};

Interval=2;
A1=[-1/75 0;0 -1/75];B1=[.878/75 .864*dt/75;1.082*dt/75 1.096/75];C1=eye(2);D1=zeros(2);
%B1=[.864/75 .878/75;1.096/75 1.082/75];
Plant1=ss(A1,B1,C1,D1);
[A,B,C,D]=ssdata(c2d(Plant1,Ts));
%Aps=A;Bps=B;Cps=C;Dps=D;
Aps=Ap{2};Bps=Bp{2};Cps=Cp{2};
%models for subsystems
%subsystem 1
A1=A(1,1);A2=A(2,2);
B11=B(1,1);B12=B(1,2);B22=B(2,2);B21=B(2,1);
C1=C(1,1);C2=C(2,2);
ywt=[1 1];
Gs=[];
Glmi=[];

uwt=[1.7986e-006  8.2675e-005];
uwt=[0 0];
uwt=[8.9908e-003  1.8289e-006];%20%
uwt=[1.8919e-005  1.0463e-002];%80%
%uwt=[1.6925e-002  5.7856e-010];
uwt=[6.7427e-004 0];
uwt=[1.02e-5 1.02e-5]
ywt1=1;ywt2=1;
%uwt1=1.4203e-003;uwt2=3.1250e-006;
uwt1=uwt(1);uwt2=uwt(2);
p1=20;p2=20;m1=5;m2=5;
Qy=eye(2);
Qu=[uwt(1) 0;0 uwt(2)];
nu1=1;ny1=1;nu2=1;ny2=1;
%create the input and output weighting matrices
wu1=[];
wu2=[];
wy1=[];
wy2=[];
for i=1:m1*ny1
    wu1=[wu1 uwt1];
end
for i=1:m2*ny2
    wu2=[wu2 uwt2];
end
for i=1:p1*ny1
    wy1=[wy1 ywt1];
end
for i=1:p2*ny2
    wy2=[wy2 ywt2];
end
Wu1=diag(wu1);
Wu2=diag(wu2);
Wy1=diag(wy1);
Wy2=diag(wy2);

EY01=eye(ny1);EY1=EY01;
for i=1:p1-1
    EY1=[EY1 EY01];
end

N1=EY1';

EY02=eye(ny2);EY2=EY02;
```

```
for i=1:p2-1
    EY2=[EY2 EY02];
end

N2=EY2';

%create the controller's paramater and Kmpcs
%controller1
nx1=size(A1,1);
nx2=size(A2,1);
Epsi1=zeros(ny1*p1,nx1);

for i=1:p1
    Epsi1((i-1)*ny1+1:i*ny1,:)=C1*A1^i;
end

%Epsi12=zeros(ny1*p1,nx2);
%for i=1:p1
%   Epsi12((i-1)*ny1+1:i*ny1,:)=C12*A2^i;
%end

Temp1=C1*B11;
GA1=Temp1;
for i=1:p1-1
    for j=1:i
        Temp1=Temp1+C1*A1^j*B11;
    end
    GA1=[GA1;Temp1];
    Temp1=C1*B11;
end

Temp12=C1*B12;
GA12=Temp12;
for i=1:p1-1
    for j=1:i
        Temp12=Temp12+C1*A1^j*B12;
    end
    GA12=[GA12;Temp12];
    Temp12=C1*B12;
end

Theta1 = zeros(ny1*p1,nu1*m1);
Theta1(1:p1*ny1,1:nu1)=GA1;

for i =2:m1
    Theta1((i-1)*ny1+1:p1*ny1,(i-1)*nu1+1:i*nu1)=Theta1(1:(p1-(i-1))*ny1,1:nu1);
end
Theta12 = zeros(ny1*p1,nu2*m1);
Theta12(1:p1*ny1,1:nu2)=GA12;

for i =2:m1
    Theta12((i-1)*ny1+1:p1*ny1,(i-1)*nu2+1:i*nu2)=Theta12(1:(p1-(i-1))*ny1,1:nu2);
end

Kfull1=inv(Theta1'*Wy1'*Wy1*Theta1+Wu1'*Wu1)*Theta1'*Wy1'*Wy1;
Kmpc1=Kfull1(1:nu1,:);

%create the controller's paramater and Kmpcs
%Controller 2
nx2=size(A2,1);

Epsi2=zeros(ny2*p2,nx2);

for i=1:p2
    Epsi2((i-1)*ny2+1:i*ny2,:)=C2*A2^i;
end

%Epsi21=zeros(ny2*p2,nx1);

%for i=1:p2
 %   Epsi21((i-1)*ny2+1:i*ny2,:)=C21*A1^i;
```

```
%end

Temp2=C2*B22;
GA2=Temp2;
for i=1:p2-1
    for j=1:i
        Temp2=Temp2+C2*A2^j*B22;
    end
    GA2=[GA2;Temp2];
    Temp2=C2*B22;
end

Temp21=C2*B21;
GA21=Temp21;
for i=1:p2-1
    for j=1:i
        Temp21=Temp21+C2*A2^j*B21;
    end
    GA21=[GA21;Temp21];
    Temp21=C2*B21;
end

Theta2 = zeros(ny2*p2,nu2*m2);
Theta2(1:p2*ny2,1:nu2)=GA2;

for i =2:m2
    Theta2((i-1)*ny2+1:p2*ny2,(i-1)*nu2+1:i*nu2)=Theta2(1:(p2-(i-1))*ny2,1:nu2);
end

Theta21 = zeros(ny2*p2,nu1*m2);
Theta21(1:p2*ny2,1:nu2)=GA21;

for i =2:m2
    Theta21((i-1)*ny2+1:p2*ny2,(i-1)*nu1+1:i*nu1)=Theta21(1:(p2-(i-1))*ny2,1:nu1);
end

Kfull2=inv(Theta2'*Wy2'*Wy2*Theta2+Wu2'*Wu2)*Theta2'*Wy2'*Wy2;
Kmpc2=Kfull2(1:nu2,:);
D1=[Kfull1 0*Kfull1;0*Kfull2 Kfull2];
D0=[0*Kfull1*Theta12 -Kfull1*Theta12;-Kfull2*Theta21 0*Kfull2*Theta21];

K=inv(eye(size(D0))-D0)*D1;
Kmpc=[K(1,:);K(m1+1,:)];
Epsi=[Epsi1 0*Epsi1;0*Epsi2 Epsi2];GA=[GA1 GA12;GA21 GA2];

nu=2;ny=2;p=p1;m=m1;nx=nx1+nx2;
Matrix1=[1 0];
for i=1:p-1
    Matrix1=[Matrix1;[1 0]];
end
Matrix2=[0 1];
for i=1:p-1
    Matrix2=[Matrix2;[0 1]];
end
Matrix=[Matrix1;Matrix2];
N2=Matrix;

J=0;
Tstop=8000;
[nx1,nx1]=size(A);
[nx2,nx2]=size(A);
x0=zeros(nx1,1);
xx=zeros(nx2,1);
x=x0;
for i=1:p
        r1((i-1)*ny1+1:i*ny1,1)=[1];
end
for i=1:p
        r2((i-1)*ny2+1:i*ny2,1)=[0];
end
r=[r1;r2];
```

```
u_old=zeros(nu,1);
YY=[];XX=[];X=[];
UU=[];
DD=[];
RR=[];
%alfa=.9;
R0=zeros(p*ny,1);
R=R0;
Rs=[];
dumax=0;

for t=0:round(Tstop/Ts)-1
    if t==1500
        for i=1:p
        r1((i-1)*ny1+1:i*ny1,1)=[-1];
        end
        for i=1:p
        r2((i-1)*ny2+1:i*ny2,1)=[0];
        end
        r=[r1;r2];

    end
    if t==3000
        for i=1:p
        r1((i-1)*ny1+1:i*ny1,1)=[1];
        end
        for i=1:p
        r2((i-1)*ny2+1:i*ny2,1)=[0];
        end
        r=[r1;r2];

    end
    if t==6000
        for i=1:p
        r1((i-1)*ny1+1:i*ny1,1)=[0];
        end
        for i=1:p
        r2((i-1)*ny2+1:i*ny2,1)=[0];
        end
        r=[r1;r2];

    end




    RR=[RR,R];
    R=alfa*R+(1-alfa)*r;

    %Rs=r;
    Rs=[Rs,r];
    XX=[XX,xx];
    y=Cps*xx;
    YY=[YY,y];
    W=N2*(y-[C1 0*C1;0*C2 C2]*x);
    du=Kmpc*(R-Epsi*x-GA*u_old-W);
    J=J+(y-[R(1);R(21)])'*Qy*(y-[R(1);R(21)])+du'*Qu*du;
    if abs(du)>dumax
        dumax=abs(du);
    end
    u=u_old+du;
    UU=[UU,u];
    u_old=u;
    x=[A1,0*A1;0*A2 A2]*x+[B11 B12;B21 B22]*u;
    xx=Aps*xx+Bps*u;
end
dumax
Time=0:Ts:Tstop-Ts;
%pause
    mm=length(R);
err1=(YY(1,:)-RR(1,:));
```

```
err2=(YY(2,:)-RR(p+1,:));
ref1=Rs(1,:);
ref2=Rs(p+1,:);

Time=0:Ts:Tstop-Ts;
plot(Time,RR(1,:),'k--',Time,RR(2,:),'r--',Time,YY(1,:),Time,YY(2,:));
legend('R1','R2','y1','y2');

if dt==1
YYnash=YY;UUnash=UU;
save Nash20 YYnash UUnash RR
end

if dt==0
YYdec=YY;UUdec=UU;
save Dec20 YYdec UUdec RR
end

pause

plot(Time,UU(1,:),Time,UU(2,:));
legend('u1','u2');
%legend('R1','R2','e1','e2');

%ref1=Rstate(1,:);
gamma=sqrt((err1*err1'+err2*err2')/(ref1*ref1'+ref2*ref2'))
J
%Gs=[Gs;gammasim];

clear
clc

alfa=0.9;
Ts=.1;
dt=1;

A=[-1.1002 .4463 0;0.6695 -1.1369 0;11.7337 0 -0.0214];B=[-.0368 0;0.0552 0;0 -.0026];
C=[0 1 0;0 0 1];D=zeros(2);


%models for subsystems
%subsystem 1
A1=A;A2=A;
B11=[B(:,1)];B12=[B(:,2)];B22=B12;B21=B11;
C1=[C(1,:)];C2=[C(2,:)];

A=[-1.1002 .4463 0;0.6695 -1.1369 0;11.7337 0 -0.0214];B=[-.0368 0;0.0552*1.005 0;0 -
.0026];
C=[0 1 0;0 0 1];D=zeros(2);

Aps=A;Bps=B;Cps=C;Dps=D;


ywt=[1 1];
Gs=[];
Glmi=[];

uwt=[0. 0.];
%uwt=[];
ywt1=1;ywt2=1;
%uwt1=1.4203e-003;uwt2=3.1250e-006;
uwt1=uwt(1);uwt2=uwt(2);
p1=10;p2=10;m1=2;m2=2;

nu1=1;ny1=1;nu2=1;ny2=1;
%create the input and output weighting matrices
wu1=[];
wu2=[];
wy1=[];
wy2=[];
for i=1:m1*ny1
```

```
    wu1=[wu1 uwt1];
end
for i=1:m2*ny2
    wu2=[wu2 uwt2];
end
for i=1:p1*ny1
    wy1=[wy1 ywt1];
end
for i=1:p2*ny2
    wy2=[wy2 ywt2];
end
Wu1=diag(wu1);
Wu2=diag(wu2);
Wy1=diag(wy1);
Wy2=diag(wy2);

EY01=eye(ny1);EY1=EY01;
for i=1:p1-1
    EY1=[EY1 EY01];
end

N1=EY1';

EY02=eye(ny2);EY2=EY02;
for i=1:p2-1
    EY2=[EY2 EY02];
end

N2=EY2';

%create the controller's paramater and Kmpcs
%controller1
nx1=size(A1,1);
nx2=size(A2,1);
Epsi1=zeros(ny1*p1,nx1);

for i=1:p1
    Epsi1((i-1)*ny1+1:i*ny1,:)=C1*A1^i;
end

%Epsi12=zeros(ny1*p1,nx2);
%for i=1:p1
%   Epsi12((i-1)*ny1+1:i*ny1,:)=C12*A2^i;
%end

Temp1=C1*B11;
GA1=Temp1;
for i=1:p1-1
    for j=1:i
        Temp1=Temp1+C1*A1^j*B11;
    end
    GA1=[GA1;Temp1];
    Temp1=C1*B11;
end

Temp12=C1*B12;
GA12=Temp12;
for i=1:p1-1
    for j=1:i
        Temp12=Temp12+C1*A1^j*B12;
    end
    GA12=[GA12;Temp12];
    Temp12=C1*B12;
end

Theta1 = zeros(ny1*p1,nu1*m1);
Theta1(1:p1*ny1,1:nu1)=GA1;

for i =2:m1
    Theta1((i-1)*ny1+1:p1*ny1,(i-1)*nu1+1:i*nu1)=Theta1(1:(p1-(i-1))*ny1,1:nu1);
end
```

```
Theta12 = zeros(ny1*p1,nu2*m1);
Theta12(1:p1*ny1,1:nu2)=GA12;

for i =2:m1
    Theta12((i-1)*ny1+1:p1*ny1,(i-1)*nu2+1:i*nu2)=Theta12(1:(p1-(i-1))*ny1,1:nu2);
end

Kfull1=inv(Theta1'*Wy1'*Wy1*Theta1+Wu1'*Wu1)*Theta1'*Wy1'*Wy1;
Kmpc1=Kfull1(1:nu1,:);

%create the controller's paramater and Kmpcs
%Controller 2
nx2=size(A2,1);

Epsi2=zeros(ny2*p2,nx2);

for i=1:p2
    Epsi2((i-1)*ny2+1:i*ny2,:)=C2*A2^i;
end

%Epsi21=zeros(ny2*p2,nx1);

%for i=1:p2
 %   Epsi21((i-1)*ny2+1:i*ny2,:)=C21*A1^i;
%end

Temp2=C2*B22;
GA2=Temp2;
for i=1:p2-1
    for j=1:i
        Temp2=Temp2+C2*A2^j*B22;
    end
    GA2=[GA2;Temp2];
    Temp2=C2*B22;
end

Temp21=C2*B21;
GA21=Temp21;
for i=1:p2-1
    for j=1:i
        Temp21=Temp21+C2*A2^j*B21;
    end
    GA21=[GA21;Temp21];
    Temp21=C2*B21;
end

Theta2 = zeros(ny2*p2,nu2*m2);
Theta2(1:p2*ny2,1:nu2)=GA2;

for i =2:m2
    Theta2((i-1)*ny2+1:p2*ny2,(i-1)*nu2+1:i*nu2)=Theta2(1:(p2-(i-1))*ny2,1:nu2);
end

Theta21 = zeros(ny2*p2,nu1*m2);
Theta21(1:p2*ny2,1:nu2)=GA21;

for i =2:m2
    Theta21((i-1)*ny2+1:p2*ny2,(i-1)*nu1+1:i*nu1)=Theta21(1:(p2-(i-1))*ny2,1:nu1);
end

Kfull2=inv(Theta2'*Wy2'*Wy2*Theta2+Wu2'*Wu2)*Theta2'*Wy2'*Wy2;
Kmpc2=Kfull2(1:nu2,:);
D1=[Kfull1 0*Kfull1;0*Kfull2 Kfull2];
D0=[0*Kfull1*Theta12 -Kfull1*Theta12;-Kfull2*Theta21 0*Kfull2*Theta21];

K=inv(eye(size(D0))-D0)*D1;
Kmpc=[K(1,:);K(m1+1,:)];
Epsi=[Epsi1 0*Epsi1;0*Epsi2 Epsi2];GA=[GA1 GA12;GA21 GA2];

nu=2;ny=2;p=p1;m=m1;nx=nx1+nx2;
Matrix1=[1 0];
```

```
for i=1:p-1
    Matrix1=[Matrix1;[1 0]];
end
Matrix2=[0 1];
for i=1:p-1
    Matrix2=[Matrix2;[0 1]];
end
Matrix=[Matrix1;Matrix2];
N2=Matrix;


Tstop=10;
[nx1,nx1]=size(A);
[nx2,nx2]=size(A);
x0=zeros(2*nx1,1);
xx=zeros(nx2,1);
x=x0;
for i=1:p1
        r1((i-1)*ny1+1:i*ny1,1)=[1];
end
for i=1:p2
        r2((i-1)*ny2+1:i*ny2,1)=[0];
end
r=[r1;r2];
u_old=zeros(nu,1);
YY=[];XX=[];X=[];
UU=[];
DD=[];
RR=[];
%alfa=.9;
R0=zeros(p*ny,1);
R=R0;
Rs=[];

for t=0:round(Tstop/Ts)-1
    if t==40
        for i=1:p
        r1((i-1)*ny1+1:i*ny1,1)=[1];
        end
        for i=1:p
        r2((i-1)*ny2+1:i*ny2,1)=[0];
        end
        r=[r1;r2];

    end



    RR=[RR,R];
    R=alfa*R+(1-alfa)*r;

    %Rs=r;
    Rs=[Rs,r];
    XX=[XX,xx];
    y=Cps*xx;
    YY=[YY,y];
    W=N2*(y-[C1 0*C1;0*C2 C2]*x);
    du=Kmpc*(R-Epsi*x-GA*u_old-W);
    u=u_old+du;
    UU=[UU,u];
    u_old=u;
    x=[A1,0*A1;0*A2 A2]*x+[B11 B12;B21 B22]*u;
    xx=Aps*xx+Bps*u;
end

Time=0:Ts:Tstop-Ts;
%pause
    mm=length(R);
err1=(YY(1,:)-RR(1,:));
err2=(YY(2,:)-RR(p+1,:));
```

```
ref1=Rs(1,:);
ref2=Rs(p+1,:);

Time=0:Ts:Tstop-Ts;
plot(Time,RR(1,:),'k--',Time,RR(2,:),'r--',Time,YY(1,:),Time,YY(2,:));
legend('R1','R2','y1','y2');

pause

plot(Time,UU(1,:),Time,UU(2,:));
legend('u1','u2');
%legend('R1','R2','e1','e2');

%ref1=Rstate(1,:);
gamma=sqrt((err1*err1'+err2*err2')/(ref1*ref1'+ref2*ref2'))

%Gs=[Gs;gammasim];

clear
clc

alfa=0;
Ts=.1;
dt=1;

A=[-1.1002 0 0.4463;11.7337 -0.0214 0;.6695 0 -1.1369];B=[-.0368 0;0 -.0026;0.0552 0];
C=[0 1 0;0 0 1];D=zeros(2);


%models for subsystems
%subsystem 1
A1=A;A2=A;
B11=[B(:,1)];B12=[B(:,2)];B22=B12;B21=B11;
C1=[C(1,:)];C2=[C(2,:)];

%A=[-1.1002 0 0.4463;11.7337 -0.0214 0;.6695 0 -1.1369];B=[-.0368 0;0 -.0026;0.0552 0];
%C=[0 0 1;0 1 0];D=zeros(2);


Aps=A;Bps=B;Cps=C;Dps=D;


ywt=[1 1];
Gs=[];
Glmi=[];

uwt=[0 0];
%uwt=[];
ywt1=1;ywt2=1;
%uwt1=1.4203e-003;uwt2=3.1250e-006;
uwt1=uwt(1);uwt2=uwt(2);
p1=10;p2=10;m1=2;m2=2;

nu1=1;ny1=1;nu2=1;ny2=1;
%create the input and output weighting matrices
wu1=[];
wu2=[];
wy1=[];
wy2=[];
for i=1:m1*ny1
    wu1=[wu1 uwt1];
end
for i=1:m2*ny2
    wu2=[wu2 uwt2];
end
for i=1:p1*ny1
    wy1=[wy1 ywt1];
end
for i=1:p2*ny2
    wy2=[wy2 ywt2];
end
```

```
Wu1=diag(wu1);
Wu2=diag(wu2);
Wy1=diag(wy1);
Wy2=diag(wy2);

EY01=eye(ny1);EY1=EY01;
for i=1:p1-1
    EY1=[EY1 EY01];
end

N1=EY1';

EY02=eye(ny2);EY2=EY02;
for i=1:p2-1
    EY2=[EY2 EY02];
end

N2=EY2';

%create the controller's paramater and Kmpcs
%controller1
nx1=size(A1,1);
nx2=size(A2,1);
Epsi1=zeros(ny1*p1,nx1);

for i=1:p1
    Epsi1((i-1)*ny1+1:i*ny1,:)=C1*A1^i;
end

%Epsi12=zeros(ny1*p1,nx2);
%for i=1:p1
%   Epsi12((i-1)*ny1+1:i*ny1,:)=C12*A2^i;
%end

Temp1=C1*B11;
GA1=Temp1;
for i=1:p1-1
    for j=1:i
        Temp1=Temp1+C1*A1^j*B11;
    end
    GA1=[GA1;Temp1];
    Temp1=C1*B11;
end

Temp12=C1*B12;
GA12=Temp12;
for i=1:p1-1
    for j=1:i
        Temp12=Temp12+C1*A1^j*B12;
    end
    GA12=[GA12;Temp12];
    Temp12=C1*B12;
end

Theta1 = zeros(ny1*p1,nu1*m1);
Theta1(1:p1*ny1,1:nu1)=GA1;

for i =2:m1
    Theta1((i-1)*ny1+1:p1*ny1,(i-1)*nu1+1:i*nu1)=Theta1(1:(p1-(i-1))*ny1,1:nu1);
end
Theta12 = zeros(ny1*p1,nu2*m1);
Theta12(1:p1*ny1,1:nu2)=GA12;

for i =2:m1
    Theta12((i-1)*ny1+1:p1*ny1,(i-1)*nu2+1:i*nu2)=Theta12(1:(p1-(i-1))*ny1,1:nu2);
end

Kfull1=inv(Theta1'*Wy1'*Wy1*Theta1+Wu1'*Wu1)*Theta1'*Wy1'*Wy1;
Kmpc1=Kfull1(1:nu1,:);

%create the controller's paramater and Kmpcs
```

```
%Controller 2
nx2=size(A2,1);

Epsi2=zeros(ny2*p2,nx2);

for i=1:p2
    Epsi2((i-1)*ny2+1:i*ny2,:)=C2*A2^i;
end

%Epsi21=zeros(ny2*p2,nx1);

%for i=1:p2
 %   Epsi21((i-1)*ny2+1:i*ny2,:)=C21*A1^i;
%end

Temp2=C2*B22;
GA2=Temp2;
for i=1:p2-1
    for j=1:i
        Temp2=Temp2+C2*A2^j*B22;
    end
    GA2=[GA2;Temp2];
    Temp2=C2*B22;
end

Temp21=C2*B21;
GA21=Temp21;
for i=1:p2-1
    for j=1:i
        Temp21=Temp21+C2*A2^j*B21;
    end
    GA21=[GA21;Temp21];
    Temp21=C2*B21;
end

Theta2 = zeros(ny2*p2,nu2*m2);
Theta2(1:p2*ny2,1:nu2)=GA2;

for i =2:m2
    Theta2((i-1)*ny2+1:p2*ny2,(i-1)*nu2+1:i*nu2)=Theta2(1:(p2-(i-1))*ny2,1:nu2);
end

Theta21 = zeros(ny2*p2,nu1*m2);
Theta21(1:p2*ny2,1:nu2)=GA21;

for i =2:m2
    Theta21((i-1)*ny2+1:p2*ny2,(i-1)*nu1+1:i*nu1)=Theta21(1:(p2-(i-1))*ny2,1:nu1);
end

Kfull2=inv(Theta2'*Wy2'*Wy2*Theta2+Wu2'*Wu2)*Theta2'*Wy2'*Wy2;
Kmpc2=Kfull2(1:nu2,:);
D1=[Kfull1 0*Kfull1;0*Kfull2 Kfull2];
D0=[0*Kfull1*Theta12 -Kfull1*Theta12;-Kfull2*Theta21 0*Kfull2*Theta21];

K=inv(eye(size(D0))-D0)*D1;
Kmpc=[K(1,:);K(m1+1,:)];
Epsi=[Epsi1 0*Epsi1;0*Epsi2 Epsi2];GA=[GA1 GA12;GA21 GA2];

nu=2;ny=2;p=p1;m=m1;nx=nx1+nx2;
Matrix1=[1 0];
for i=1:p-1
    Matrix1=[Matrix1;[1 0]];
end
Matrix2=[0 1];
for i=1:p-1
    Matrix2=[Matrix2;[0 1]];
end
Matrix=[Matrix1;Matrix2];
N2=Matrix;
```

```
Tstop=100;
[nx1,nx1]=size(A);
[nx2,nx2]=size(A);
x0=zeros(2*nx1,1);
xx=zeros(nx2,1);
x=x0;
for i=1:p1
        r1((i-1)*ny1+1:i*ny1,1)=[0];
end
for i=1:p2
        r2((i-1)*ny2+1:i*ny2,1)=[1];
end
r=[r1;r2];
u_old=zeros(nu,1);
YY=[];XX=[];X=[];
UU=[];
DD=[];
RR=[];
%alfa=.9;
R0=zeros(p*ny,1);
R=R0;
Rs=[];

for t=0:round(Tstop/Ts)-1
    if t==40
        for i=1:p
        r1((i-1)*ny1+1:i*ny1,1)=[0];
        end
        for i=1:p
        r2((i-1)*ny2+1:i*ny2,1)=[1];
        end
        r=[r1;r2];

    end




    RR=[RR,R];
    R=alfa*R+(1-alfa)*r;

    %Rs=r;
    Rs=[Rs,r];
    XX=[XX,xx];
    y=Cps*xx;
    YY=[YY,y];
    W=N2*(y-[C1 0*C1;0*C2 C2]*x);
    du=Kmpc*(R-Epsi*x-GA*u_old-W);
    u=u_old+du;
    UU=[UU,u];
    u_old=u;
    x=[A1,0*A1;0*A2 A2]*x+[B11 B12;B21 B22]*u;
    xx=Aps*xx+Bps*u;
end

Time=0:Ts:Tstop-Ts;
%pause
    mm=length(R);
err1=(YY(1,:)-RR(1,:));
err2=(YY(2,:)-RR(p+1,:));
ref1=Rs(1,:);
ref2=Rs(p+1,:);

Time=0:Ts:Tstop-Ts;
plot(Time,RR(1,:),'k--',Time,RR(2,:),'r--',Time,YY(1,:),Time,YY(2,:));
legend('R1','R2','y1','y2');

pause

plot(Time,UU(1,:),Time,UU(2,:));
legend('u1','u2');
```

```
%legend('R1','R2','e1','e2');

%ref1=Rstate(1,:);
gamma=sqrt((err1*err1'+err2*err2')/(ref1*ref1'+ref2*ref2'))

%Gs=[Gs;gammasim];

function [tmin]=CheckStability(A1,A2,A3,A4)

[ns,ns]=size(A1);
[ns,ns]=size(A1);
setlmis([])
Po=lmivar(1,[ns 1]);%Po is symmetric block diagonal
Gamm=lmivar(1,[1 1]);%

%LMI#1
lmiterm([1 1 1 Po],A1',A1);%A'*Po*A
lmiterm([1 1 1 Po],-1,1);%-Po

%LMI#2
lmiterm([2 1 1 Po],A2',A2);%A'*Po*A
lmiterm([2 1 1 Po],-1,1);%-Po

%LMI#3
lmiterm([3 1 1 Po],A3',A3);%A'*Po*A
lmiterm([3 1 1 Po],-1,1);%-Po

%LMI#4
lmiterm([4 1 1 Po],A4',A4);%A'*Po*A
lmiterm([4 1 1 Po],-1,1);%-Po

%LMI#5
lmiterm([5 1 1 Po],-1,1);

LMIsys=getlmis;

[tmin,xfeas]=feasp(LMIsys,[1.0*exp(-4) 100 1e5 40 1]);

function gamma=Glmi(lambda)
%if lambda(1)<0
%    gamma=1e6;
%    return;
%end
%if lambda(2)<0
%    gamma=1e6;
%    return;
%end
%if lambda(3)<0
%    gamma=1e6;
%    return;
%end
%if lambda(4)<0
%    gamma=1e6;
%    return;
%end
%if lambda(5)<0
%    gamma=1e6;
%    return;
%end

alfa=0.99;

Ts=.1;

us=[4654.2;8;570;600;282.82;15;15];
xs=[2.0518e-001
  1.9460e-001
  3.1250e+002
  4.3721e+002
  4.9515e-001
  2.5910e-001
```

```
  3.7454e-001
  1.9175e-001
  1.0492e-001
  6.3679e-002
  4.4089e-002
  3.4784e-002
  1.8210e-002
  9.5183e-003
  4.9598e-003
  2.5691e-003
  2.6439e-002
  2.2451e+002
  3.6117e+002]';
tx=diag(xs);
tu=diag(us);
%plant vertices
load point1010;

A=inv(tx)*A*tx;B=inv(tx)*[B BD]*tu;C=C*tx;
Plant=ss(A,B,C,zeros(5,7));

[Ap{1},BB,Cp{1},D]=ssdata(c2d(Plant,Ts));
Bp{1}=BB(:,1:5);Bd{1}=BB(:,6:7);



load point1020;
A=inv(tx)*A*tx;B=inv(tx)*[B BD]*tu;C=C*tx;

Plant=ss(A,B,C,zeros(5,7));
[Ap{2},BB,Cp{2},D]=ssdata(c2d(Plant,Ts));
Bp{2}=BB(:,1:5);Bd{2}=BB(:,6:7);

load point2010;
A=inv(tx)*A*tx;B=inv(tx)*[B BD]*tu;C=C*tx;

Plant=ss(A,B,C,zeros(5,7));
[Ap{3},BB,Cp{3},D]=ssdata(c2d(Plant,Ts));
Bp{3}=BB(:,1:5);Bd{3}=BB(:,6:7);


load point2020;
A=inv(tx)*A*tx;B=inv(tx)*[B BD]*tu;C=C*tx;

Plant=ss(A,B,C,zeros(5,7));
[Ap{4},BB,Cp{4},D]=ssdata(c2d(Plant,Ts));
Bp{4}=BB(:,1:5);Bd{4}=BB(:,6:7);


us=[4654.2;8;570;600;282.82];
tu=diag(us);

%nominal point
load nominal;
dt=0;
index=1;
if index==1
% unit based
% unit based
%A=[A(1:4,1:4) dt*A(1:4,5:18) A(1:4,19);dt*A(5:6,1:4) A(5:6,5:6) dt*A(5:6,7:19);...
%   dt*A(7:11,1:6) A(7:11,7:11) dt*A(7:11,12:19);...
%   dt*A(12:16,1:11) A(12:16,12:16) dt*A(12:16,17:19);...
%   dt*A(17:18,1:16) A(17:18,17:18) dt*A(17:18,19);...
%   dt*A(19,1:18) A(19,19)];
%B(1:19,1)=[B(1:4,1);dt*B(5:19,1)];
%B(1:19,2)=[dt*B(1:4,2);B(5:6,2);dt*B(7:19,2)];
%B(1:19,3)=[dt*B(1:6,3);B(7:11,3);dt*B(12:19,3)];
%B(1:19,4)=[dt*B(1:11,4);B(12:16,4);dt*B(17:19,4)];
%B(1:19,5)=[dt*B(1:16,5);B(17:18,5);dt*B(19,5)];
% A=[A(1:4,1:4) dt*A(1:4,5:18) A(1:4,19);dt*A(5:6,1:4) b5:6,5:6) dt*A(5:6,7:19);...
%    dt*A(7:16,1:6) A(7:16,7:16) dt*A(7:16,17:19);...
```

```
%    dt*A(17:18,1:16) A(17:18,17:18) dt*A(17:18,19);...
%    dt*A(19,1:18) A(19,19)];
% B(1:19,1)=[B(1:4,1);dt*B(5:19,1)];
% B(1:19,2)=[dt*B(1:4,2);B(5:6,2);dt*B(7:19,2)];
% B(1:19,3)=[dt*B(1:6,3);B(7:16,3);dt*B(17:19,3)];
% B(1:19,4)=[dt*B(1:6,4);B(7:16,4);dt*B(17:19,4)];
% B(1:19,5)=[dt*B(1:16,5);B(17:18,5);dt*B(19,5)];
end

if index==2
%dau7
A=[A(1:4,1:4) zeros(4,2) A(1:4,7:16) zeros(4,2) A(1:4,19); zeros(2,4) A(5:6,5:6)
zeros(2,10) A(5:6,17:18) zeros(2,1); A(7:16,1:4) zeros(10,2) A(7:16,7:16) zeros(10,2)
A(7:16,19); zeros(2,4) A(17:18,5:6) zeros(2,10) A(17:18,17:18) zeros(2,1); A(19,1:4)
zeros(1,2) A(19,7:16) zeros(1,2) A(19,19)];
B=[B(1:4,1) zeros(4,1) B(1:4,3:4) zeros(4,1); zeros(2,1) B(5:6,2) zeros(2,2) B(5:6,5);
B(7:16,1) zeros(10,1) B(7:16,3:4) zeros(10,1); zeros(2,1) B(17:18,2) zeros(2,2)
B(17:18,5); B(19,1) zeros(1,1) B(19,3:4) zeros(1,1)];
end

if index==3
%dau8
A=[A(1:4,1:4) zeros(4,14) A(1:4,19); zeros(14,4) A(5:18,5:18) zeros(14,1); A(19,1:4)
zeros(1,14) A(19,19)];
B=[B(1:4,1) zeros(4,4); zeros(14,1) B(5:18,2:5); zeros(1,5)];
end

A=inv(tx)*A*tx;B=inv(tx)*B*tu;C=C*tx;
%B=B(:,[1 2 3 5 4]);
Plant=ss(A,B,C,zeros(5,5));
[A,B,C,D]=ssdata(c2d(Plant,Ts));
A=[A(1:4,1:4) dt*A(1:4,5:18) A(1:4,19);dt*A(5:6,1:4) A(5:6,5:6) dt*A(5:6,7:19);...
    dt*A(7:16,1:6) A(7:16,7:16) dt*A(7:16,17:19);...
    dt*A(17:18,1:16) A(17:18,17:18) dt*A(17:18,19);...
    dt*A(19,1:18) A(19,19)];
B(1:19,1)=[B(1:4,1);dt*B(5:19,1)];
B(1:19,2)=[dt*B(1:4,2);B(5:6,2);dt*B(7:19,2)];
B(1:19,3)=[dt*B(1:6,3);B(7:16,3);dt*B(17:19,3)];
B(1:19,4)=[dt*B(1:6,4);B(7:16,4);dt*B(17:19,4)];
B(1:19,5)=[dt*B(1:16,5);B(17:18,5);dt*B(19,5)];



Interval=4;

ywt=[1 1 1 1 1];

%uwt=10*[0.0002    0.3711    0.5999    0.2408    0.1783];
%uwt=[0 .4 .6 .2 .2];
uwt=[lambda(1) lambda(2) lambda(3) lambda(4) lambda(5)]

p=10;
m=2;

nu=5;ny=5;nd=2;

weiu=uwt;
weiy=ywt;

EU0=eye(nu);EU=EU0;
for i=1:m-1
    EU=[EU EU0];
end
Wu=spdiags([weiu*EU]',[0],m*nu,m*nu);

%Wy pny*pny
EY0=eye(ny);EY=EY0;
for i=1:p-1
    EY=[EY EY0];
end
Wy=spdiags([weiy*EY]',[0],p*ny,p*ny);
```

```
N2=EY';

[nx,nx]=size(A);

Epsi=zeros(ny*p,nx);

for i=1:p
    Epsi((i-1)*ny+1:i*ny,:)=C*A^i;
end

Temp=C*B;
GA=Temp;
for i=1:p-1
    for j=1:i
        Temp=Temp+C*A^j*B;
    end
    GA=[GA;Temp];
    Temp=C*B;
end

Theta = zeros(ny*p,nu*m);
Theta(1:p*ny,1:nu)=GA;

for i =2:m
    Theta((i-1)*ny+1:p*ny,(i-1)*nu+1:i*nu)=Theta(1:(p-(i-1))*ny,1:nu);
end

Kfull=inv(Theta'*Wy'*Wy*Theta+Wu'*Wu)*Theta'*Wy'*Wy;
Kmpc=Kfull(1:nu,:);

%kk=0.001*ones(nx,ny);
%NN=C*kk;
%for i=1:p-1
%    NN=[NN;C*A^i*kk];
%end
%N2=NN;

for i=1:Interval

%process Xk+1
A11=Ap{i}-Bp{i}*Kmpc*N2*Cp{i};
A12=Bp{i}*Kmpc*(N2*C-Epsi);
A13=Bp{i}-Bp{i}*Kmpc*GA;
A14=Bd{i};

%nominal model Xk+1
A21=-B*Kmpc*N2*Cp{i};
A22=A+B*Kmpc*(N2*C-Epsi);
A23=B-B*Kmpc*GA;
A24=zeros(nx,nd);

%Uk
A31=-Kmpc*N2*Cp{i};
A32=Kmpc*(N2*C-Epsi);
A33=eye(nu,nu)-Kmpc*GA;
A34=zeros(nu,nd);

%dk+1
A41=zeros(nd,nx);
A42=zeros(nd,nx);
A43=zeros(nd,nu);
A44=(alfa)*[1 0;0 1];
%aa=[A11 A12 A13;A21 A22 A23;A31 A32 A33];
%AA=[A11 zeros(2,2) A12 A13;zeros(2,2) A11 A12 A13;.5*A21 .5*A21 A22 A23;A31 A31 A32
A33];

Asys{i}=[A11 A12 A13 A14;A21 A22 A23 A24;A31 A32 A33 A34;A41 A42 A43 A44];
%Asys{i}=[A11 A12 A13;A21 A22 A23;A31 A32 A33];

%Asys=AA;
```

```
B1=zeros(nx,nd);
B2=zeros(nx,nd);
B3=zeros(nu,nd);
B4=(1-alfa)*[1 0;0 1];
%bb=[B1;B2;B3];
Bsys{i}=[B1;B2;B3;B4];

%Csys=[-Cp*(Ap-Bp*Kmpc*N2*Cp),-Cp*(Bp*Kmpc*(N2*C-Epsi)),-Cp*(Bp-
Bp*Kmpc*GA),alfa*eye(ny,ny)-Cp*Bp*Kmpc*alfa*Matrix];
%Csys=[-Cp*(Ap-Bp*(Kmpc*N2*Cp)),-Cp*(Bp*Kmpc*(N2*C-Epsi)),-Cp*(Bp-Bp*Kmpc*GA)];

Csys{i}=[-Cp{i},zeros(ny,nx),zeros(ny,nu),zeros(ny,nd)];
%Dsys=(1-alfa)*eye(ny,ny)-Cp*Bp*Kmpc*(1-alfa)*Matrix;
%Dsys=(1-alfa)*eye(ny,ny);
Dsys=zeros(ny,nd);
MPCsys{i}=ss(Asys{i},Bsys{i},Csys{i},Dsys,Ts);
Poles{i}=pole(MPCsys{i});
%eig(Asys{i})

end

[sys,g,t,ti] = balreal(MPCsys{1});  % Compute balanced realization
elim = (g<1e-6);         % Small entries of g are negligible states
rsys1 = modred(sys,elim);

[sys,g] = balreal(MPCsys{2});  % Compute balanced realization
rsys2 = modred(sys,elim);

[sys,g] = balreal(MPCsys{3});  % Compute balanced realization
rsys3 = modred(sys,elim);

[sys,g] = balreal(MPCsys{4});  % Compute balanced realization
rsys4 = modred(sys,elim);


A1=rsys1.a;
A2=rsys2.a;
A3=rsys3.a;
A4=rsys4.a;
B1=rsys1.b;
B2=rsys2.b;
B3=rsys3.b;
B4=rsys4.b;
C1=rsys1.c;
C2=rsys2.c;
C3=rsys3.c;
C4=rsys4.c;
D1=rsys1.d;
D2=rsys2.d;
D3=rsys3.d;
D4=rsys4.d;

[tmin]=CheckStability(A1,A2,A3,A4)
if tmin>0
    gamma=1e12
    return
end
%tic;
[ns,ns]=size(A1);
setlmis([])
Po=lmivar(1,[ns 1]);%Po is symmetric block diagonal
Gamm=lmivar(1,[1 1]);%

%LMI#1
lmiterm([1 1 1 Po],A1',A1);%A'*Po*A
lmiterm([1 1 1 Po],-1,1);%-Po
lmiterm([1 1 2 Po],A1',B1);%A'*Po*B
lmiterm([1 1 3 0],C1');%C'
lmiterm([1 2 2 Po],B1',B1);%B'*Po*B
lmiterm([1 2 2 Gamm],-1,1);%-gamma2I
lmiterm([1 2 3 0],D1');%D'
```

```
lmiterm([1 3 3 0],-1);%1

%LMI#2
lmiterm([2 1 1 Po],A2',A2);%A'*Po*A
lmiterm([2 1 1 Po],-1,1);%-Po
lmiterm([2 1 2 Po],A2',B2);%A'*Po*B
lmiterm([2 1 3 0],C2');%C'
lmiterm([2 2 2 Po],B2',B2);%B'*Po*B
lmiterm([2 2 2 Gamm],-1,1);%-gamma2I
lmiterm([2 2 3 0],D2');%D'
lmiterm([2 3 3 0],-1);%1

%LMI#3
lmiterm([3 1 1 Po],A3',A3);%A'*Po*A
lmiterm([3 1 1 Po],-1,1);%-Po
lmiterm([3 1 2 Po],A3',B3);%A'*Po*B
lmiterm([3 1 3 0],C3');%C'
lmiterm([3 2 2 Po],B3',B3);%B'*Po*B
lmiterm([3 2 2 Gamm],-1,1);%-gamma2I
lmiterm([3 2 3 0],D3');%D'
lmiterm([3 3 3 0],-1);%1

%LMI#4
lmiterm([4 1 1 Po],A4',A4);%A'*Po*A
lmiterm([4 1 1 Po],-1,1);%-Po
lmiterm([4 1 2 Po],A4',B4);%A'*Po*B
lmiterm([4 1 3 0],C4');%C'
lmiterm([4 2 2 Po],B4',B4);%B'*Po*B
lmiterm([4 2 2 Gamm],-1,1);%-gamma2I
lmiterm([4 2 3 0],D4');%D'
lmiterm([4 3 3 0],-1);%1

%LMI#5
lmiterm([5 1 1 Po],-1,1);

LMIsys=getlmis;
CC=mat2dec(LMIsys,zeros(ns),1);
%CC'
%CC=zeros(1,947);CC(1)=1;
[gam,xopt]=mincx(LMIsys,CC,[1.0*exp(-4) 100 1e5 40 1]);
%Qm = dec2mat(LMIsys,xopt,Gamm)
gamma=sqrt(gam)
%toc;
%t=toc;


function gamma=Glmi(lambda)
%if lambda(1)<0
%    gamma=1e6;
%    return;
%end
%if lambda(2)<0
%    gamma=1e6;
%    return;
%end
%if lambda(3)<0
%    gamma=1e6;
%    return;
%end
%if lambda(4)<0
%    gamma=1e6;
%    return;
%end
%if lambda(5)<0
%    gamma=1e6;
%    return;
%end

alfa=0.99;

Ts=.1;
```

```
us=[4654.2;8;570;600;282.82;15;15];
xs=[2.0518e-001
  1.9460e-001
  3.1250e+002
  4.3721e+002
  4.9515e-001
  2.5910e-001
  3.7454e-001
  1.9175e-001
  1.0492e-001
  6.3679e-002
  4.4089e-002
  3.4784e-002
  1.8210e-002
  9.5183e-003
  4.9598e-003
  2.5691e-003
  2.6439e-002
  2.2451e+002
  3.6117e+002]';
tx=diag(xs);
tu=diag(us);
%plant vertices
load point1010;

A=inv(tx)*A*tx;B=inv(tx)*[B BD]*tu;C=C*tx;
Plant=ss(A,B,C,zeros(5,7));

[Ap{1},BB,Cp{1},D]=ssdata(c2d(Plant,Ts));
Bp{1}=BB(:,1:5);Bd{1}=BB(:,6:7);



load point1020;
A=inv(tx)*A*tx;B=inv(tx)*[B BD]*tu;C=C*tx;

Plant=ss(A,B,C,zeros(5,7));
[Ap{2},BB,Cp{2},D]=ssdata(c2d(Plant,Ts));
Bp{2}=BB(:,1:5);Bd{2}=BB(:,6:7);

load point2010;
A=inv(tx)*A*tx;B=inv(tx)*[B BD]*tu;C=C*tx;

Plant=ss(A,B,C,zeros(5,7));
[Ap{3},BB,Cp{3},D]=ssdata(c2d(Plant,Ts));
Bp{3}=BB(:,1:5);Bd{3}=BB(:,6:7);


load point2020;
A=inv(tx)*A*tx;B=inv(tx)*[B BD]*tu;C=C*tx;

Plant=ss(A,B,C,zeros(5,7));
[Ap{4},BB,Cp{4},D]=ssdata(c2d(Plant,Ts));
Bp{4}=BB(:,1:5);Bd{4}=BB(:,6:7);


us=[4654.2;8;570;600;282.82];
tu=diag(us);

%nominal point
load nominal;
dt=0;
index=1;
if index==1
% unit based
% unit based
%A=[A(1:4,1:4) dt*A(1:4,5:18) A(1:4,19);dt*A(5:6,1:4) A(5:6,5:6) dt*A(5:6,7:19);...
%   dt*A(7:11,1:6) A(7:11,7:11) dt*A(7:11,12:19);...
%   dt*A(12:16,1:11) A(12:16,12:16) dt*A(12:16,17:19);...
%   dt*A(17:18,1:16) A(17:18,17:18) dt*A(17:18,19);...
```

```
%   dt*A(19,1:18) A(19,19)];
%B(1:19,1)=[B(1:4,1);dt*B(5:19,1)];
%B(1:19,2)=[dt*B(1:4,2);B(5:6,2);dt*B(7:19,2)];
%B(1:19,3)=[dt*B(1:6,3);B(7:11,3);dt*B(12:19,3)];
%B(1:19,4)=[dt*B(1:11,4);B(12:16,4);dt*B(17:19,4)];
%B(1:19,5)=[dt*B(1:16,5);B(17:18,5);dt*B(19,5)];
% A=[A(1:4,1:4) dt*A(1:4,5:18) A(1:4,19);dt*A(5:6,1:4) b5:6,5:6) dt*A(5:6,7:19);...
%    dt*A(7:16,1:6) A(7:16,7:16) dt*A(7:16,17:19);...
%    dt*A(17:18,1:16) A(17:18,17:18) dt*A(17:18,19);...
%    dt*A(19,1:18) A(19,19)];
% B(1:19,1)=[B(1:4,1);dt*B(5:19,1)];
% B(1:19,2)=[dt*B(1:4,2);B(5:6,2);dt*B(7:19,2)];
% B(1:19,3)=[dt*B(1:6,3);B(7:16,3);dt*B(17:19,3)];
% B(1:19,4)=[dt*B(1:6,4);B(7:16,4);dt*B(17:19,4)];
% B(1:19,5)=[dt*B(1:16,5);B(17:18,5);dt*B(19,5)];
end

if index==2
%dau7
A=[A(1:4,1:4) zeros(4,2) A(1:4,7:16) zeros(4,2) A(1:4,19); zeros(2,4) A(5:6,5:6)
zeros(2,10) A(5:6,17:18) zeros(2,1); A(7:16,1:4) zeros(10,2) A(7:16,7:16) zeros(10,2)
A(7:16,19); zeros(2,4) A(17:18,5:6) zeros(2,10) A(17:18,17:18) zeros(2,1); A(19,1:4)
zeros(1,2) A(19,7:16) zeros(1,2) A(19,19)];
B=[B(1:4,1) zeros(4,1) B(1:4,3:4) zeros(4,1); zeros(2,1) B(5:6,2) zeros(2,2) B(5:6,5);
B(7:16,1) zeros(10,1) B(7:16,3:4) zeros(10,1); zeros(2,1) B(17:18,2) zeros(2,2)
B(17:18,5); B(19,1) zeros(1,1) B(19,3:4) zeros(1,1)];
end

if index==3
%dau8
A=[A(1:4,1:4) zeros(4,14) A(1:4,19); zeros(14,4) A(5:18,5:18) zeros(14,1); A(19,1:4)
zeros(1,14) A(19,19)];
B=[B(1:4,1) zeros(4,4); zeros(14,1) B(5:18,2:5); zeros(1,5)];
end

A=inv(tx)*A*tx;B=inv(tx)*B*tu;C=C*tx;
%B=B(:,[1 2 3 5 4]);
Plant=ss(A,B,C,zeros(5,5));
[A,B,C,D]=ssdata(c2d(Plant,Ts));
A=[A(1:4,1:4) dt*A(1:4,5:18) A(1:4,19);dt*A(5:6,1:4) A(5:6,5:6) dt*A(5:6,7:19);...
    dt*A(7:16,1:6) A(7:16,7:16) dt*A(7:16,17:19);...
    dt*A(17:18,1:16) A(17:18,17:18) dt*A(17:18,19);...
    dt*A(19,1:18) A(19,19)];
B(1:19,1)=[B(1:4,1);dt*B(5:19,1)];
B(1:19,2)=[dt*B(1:4,2);B(5:6,2);dt*B(7:19,2)];
B(1:19,3)=[dt*B(1:6,3);B(7:16,3);dt*B(17:19,3)];
B(1:19,4)=[dt*B(1:6,4);B(7:16,4);dt*B(17:19,4)];
B(1:19,5)=[dt*B(1:16,5);B(17:18,5);dt*B(19,5)];



Interval=4;

ywt=[1 1 1 1 1];

%uwt=10*[0.0002    0.3711    0.5999    0.2408    0.1783];
%uwt=[0 .4 .6 .2 .2];
uwt=[lambda(1) lambda(2) lambda(3) lambda(4) lambda(5)]

p=10;
m=2;

nu=5;ny=5;nd=2;

weiu=uwt;
weiy=ywt;

EU0=eye(nu);EU=EU0;
for i=1:m-1
    EU=[EU EU0];
end
```

```
Wu=spdiags([weiu*EU]',[0],m*nu,m*nu);

%Wy pny*pny
EY0=eye(ny);EY=EY0;
for i=1:p-1
    EY=[EY EY0];
end
Wy=spdiags([weiy*EY]',[0],p*ny,p*ny);

N2=EY';

[nx,nx]=size(A);

Epsi=zeros(ny*p,nx);

for i=1:p
    Epsi((i-1)*ny+1:i*ny,:)=C*A^i;
end

Temp=C*B;
GA=Temp;
for i=1:p-1
    for j=1:i
        Temp=Temp+C*A^j*B;
    end
    GA=[GA;Temp];
    Temp=C*B;
end

Theta = zeros(ny*p,nu*m);
Theta(1:p*ny,1:nu)=GA;

for i =2:m
    Theta((i-1)*ny+1:p*ny,(i-1)*nu+1:i*nu)=Theta(1:(p-(i-1))*ny,1:nu);
end

Kfull=inv(Theta'*Wy'*Wy*Theta+Wu'*Wu)*Theta'*Wy'*Wy;
Kmpc=Kfull(1:nu,:);

%kk=0.001*ones(nx,ny);
%NN=C*kk;
%for i=1:p-1
%    NN=[NN;C*A^i*kk];
%end
%N2=NN;

for i=1:Interval

%process Xk+1
A11=Ap{i}-Bp{i}*Kmpc*N2*Cp{i};
A12=Bp{i}*Kmpc*(N2*C-Epsi);
A13=Bp{i}-Bp{i}*Kmpc*GA;
A14=Bd{i};

%nominal model Xk+1
A21=-B*Kmpc*N2*Cp{i};
A22=A+B*Kmpc*(N2*C-Epsi);
A23=B-B*Kmpc*GA;
A24=zeros(nx,nd);

%Uk
A31=-Kmpc*N2*Cp{i};
A32=Kmpc*(N2*C-Epsi);
A33=eye(nu,nu)-Kmpc*GA;
A34=zeros(nu,nd);

%dk+1
A41=zeros(nd,nx);
A42=zeros(nd,nx);
A43=zeros(nd,nu);
A44=(alfa)*[1 0;0 1];
```

```
%aa=[A11 A12 A13;A21 A22 A23;A31 A32 A33];
%AA=[A11 zeros(2,2) A12 A13;zeros(2,2) A11 A12 A13;.5*A21 .5*A21 A22 A23;A31 A31 A32
A33];

Asys{i}=[A11 A12 A13 A14;A21 A22 A23 A24;A31 A32 A33 A34;A41 A42 A43 A44];
%Asys{i}=[A11 A12 A13;A21 A22 A23;A31 A32 A33];

%Asys=AA;
B1=zeros(nx,nd);
B2=zeros(nx,nd);
B3=zeros(nu,nd);
B4=(1-alfa)*[1 0;0 1];
%bb=[B1;B2;B3];
Bsys{i}=[B1;B2;B3;B4];

%Csys=[-Cp*(Ap-Bp*Kmpc*N2*Cp),-Cp*(Bp*Kmpc*(N2*C-Epsi)),-Cp*(Bp-
Bp*Kmpc*GA),alfa*eye(ny,ny)-Cp*Bp*Kmpc*alfa*Matrix];
%Csys=[-Cp*(Ap-Bp*(Kmpc*N2*Cp)),-Cp*(Bp*Kmpc*(N2*C-Epsi)),-Cp*(Bp-Bp*Kmpc*GA)];

Csys{i}=[-Cp{i},zeros(ny,nx),zeros(ny,nu),zeros(ny,nd)];
%Dsys=(1-alfa)*eye(ny,ny)-Cp*Bp*Kmpc*(1-alfa)*Matrix;
%Dsys=(1-alfa)*eye(ny,ny);
Dsys=zeros(ny,nd);
MPCsys{i}=ss(Asys{i},Bsys{i},Csys{i},Dsys,Ts);
Poles{i}=pole(MPCsys{i});
%eig(Asys{i})

end

[sys,g,t,ti] = balreal(MPCsys{1});  % Compute balanced realization
elim = (g<1e-6);         % Small entries of g are negligible states
rsys1 = modred(sys,elim);

[sys,g] = balreal(MPCsys{2});  % Compute balanced realization
rsys2 = modred(sys,elim);

[sys,g] = balreal(MPCsys{3});  % Compute balanced realization
rsys3 = modred(sys,elim);

[sys,g] = balreal(MPCsys{4});  % Compute balanced realization
rsys4 = modred(sys,elim);


A1=rsys1.a;
A2=rsys2.a;
A3=rsys3.a;
A4=rsys4.a;
B1=rsys1.b;
B2=rsys2.b;
B3=rsys3.b;
B4=rsys4.b;
C1=rsys1.c;
C2=rsys2.c;
C3=rsys3.c;
C4=rsys4.c;
D1=rsys1.d;
D2=rsys2.d;
D3=rsys3.d;
D4=rsys4.d;

[tmin]=CheckStability(A1,A2,A3,A4)
if tmin>0
    gamma=1e12
    return
end
%tic;
[ns,ns]=size(A1);
setlmis([])
Po=lmivar(1,[ns 1]);%Po is symmetric block diagonal
Gamm=lmivar(1,[1 1]);%
```

```
%LMI#1
lmiterm([1 1 1 Po],A1',A1);%A'*Po*A
lmiterm([1 1 1 Po],-1,1);%-Po
lmiterm([1 1 2 Po],A1',B1);%A'*Po*B
lmiterm([1 1 3 0],C1');%C'
lmiterm([1 2 2 Po],B1',B1);%B'*Po*B
lmiterm([1 2 2 Gamm],-1,1);%-gamma2I
lmiterm([1 2 3 0],D1');%D'
lmiterm([1 3 3 0],-1);%1

%LMI#2
lmiterm([2 1 1 Po],A2',A2);%A'*Po*A
lmiterm([2 1 1 Po],-1,1);%-Po
lmiterm([2 1 2 Po],A2',B2);%A'*Po*B
lmiterm([2 1 3 0],C2');%C'
lmiterm([2 2 2 Po],B2',B2);%B'*Po*B
lmiterm([2 2 2 Gamm],-1,1);%-gamma2I
lmiterm([2 2 3 0],D2');%D'
lmiterm([2 3 3 0],-1);%1

%LMI#3
lmiterm([3 1 1 Po],A3',A3);%A'*Po*A
lmiterm([3 1 1 Po],-1,1);%-Po
lmiterm([3 1 2 Po],A3',B3);%A'*Po*B
lmiterm([3 1 3 0],C3');%C'
lmiterm([3 2 2 Po],B3',B3);%B'*Po*B
lmiterm([3 2 2 Gamm],-1,1);%-gamma2I
lmiterm([3 2 3 0],D3');%D'
lmiterm([3 3 3 0],-1);%1

%LMI#4
lmiterm([4 1 1 Po],A4',A4);%A'*Po*A
lmiterm([4 1 1 Po],-1,1);%-Po
lmiterm([4 1 2 Po],A4',B4);%A'*Po*B
lmiterm([4 1 3 0],C4');%C'
lmiterm([4 2 2 Po],B4',B4);%B'*Po*B
lmiterm([4 2 2 Gamm],-1,1);%-gamma2I
lmiterm([4 2 3 0],D4');%D'
lmiterm([4 3 3 0],-1);%1

%LMI#5
lmiterm([5 1 1 Po],-1,1);

LMIsys=getlmis;
CC=mat2dec(LMIsys,zeros(ns),1);
%CC'
%CC=zeros(1,947);CC(1)=1;
[gam,xopt]=mincx(LMIsys,CC,[1.0*exp(-4) 100 1e5 40 1]);
%Qm = dec2mat(LMIsys,xopt,Gamm)
gamma=sqrt(gam)
%toc;
%t=toc;
```

Key MATLAB codes used in Chapter 4

```
k11=32.63;tau11=(99.6*.35);theta11=(99.6+.35);
k12=-33.89;tau12=(98.02*.42);theta12=(98.02+.42);
k21=34.84;tau21=(110.5*.03);theta21=110.5+.03;
k22=-18.85;tau22=75.43*.3;theta22=75.43+.3;

Ac=[0 1 0 0 0 0 0 0
    -1/tau11 -theta11/tau11 0 0 0 0 0 0
    0 0 0 1 0 0 0 0
    0 0 -1/tau12 -theta12/tau12 0 0 0 0
    0 0 0 0 0 1 0 0
    0 0 0 0 -1/tau21 -theta21/tau21 0 0
    0 0 0 0 0 0 0 1
    0 0 0 0 0 0 -1/tau22 -theta22/tau22];
```

```
Bc=[0 0
    k11/tau11 0
    0 0
    0 k12/tau12
    0 0
    k21/tau21 0
    0 0
    0 k22/tau22];

C=[1 0 1 0 0 0 0 0;0 0 0 0 1 0 1 0];
D=[0 0;0 0];
plant2=(c2d(ss(Ac,Bc,C,D),Ts));
Pv1=plant2;
[A1,B1,C,D]=ssdata(plant2); %state-space matrices



k11=10*32.63;tau11=(99.6*.35);theta11=(99.6+.35);
k12=10*-33.89;tau12=(98.02*.42);theta12=(98.02+.42);
k21=10*34.84;tau21=(110.5*.03);theta21=110.5+.03;
k22=10*-18.85;tau22=75.43*.3;theta22=75.43+.3;

Ac=[0 1 0 0 0 0 0 0
    -1/tau11 -theta11/tau11 0 0 0 0 0 0
    0 0 0 1 0 0 0 0
    0 0 -1/tau12 -theta12/tau12 0 0 0 0
    0 0 0 0 0 1 0 0
    0 0 0 0 -1/tau21 -theta21/tau21 0 0
    0 0 0 0 0 0 0 1
    0 0 0 0 0 0 -1/tau22 -theta22/tau22];

Bc=[0 0
    k11/tau11 0
    0 0
    0 k12/tau12
    0 0
    k21/tau21 0
    0 0
    0 k22/tau22];

C=[1 0 1 0 0 0 0 0;0 0 0 0 1 0 1 0];
D=[0 0;0 0];
plant2=(c2d(ss(Ac,Bc,C,D),Ts));
Pv2=plant2;

[A2,B2,C,D]=ssdata(plant2); %state-space matrices



k11=5*32.63;tau11=(99.6*.35);theta11=(99.6+.35);
k12=5*-33.89;tau12=(98.02*.42);theta12=(98.02+.42);
k21=5*34.84;tau21=(110.5*.03);theta21=110.5+.03;
k22=5*-18.85;tau22=75.43*.3;theta22=75.43+.3;

Ac=[0 1 0 0 0 0 0 0
    -1/tau11 -theta11/tau11 0 0 0 0 0 0
    0 0 0 1 0 0 0 0
    0 0 -1/tau12 -theta12/tau12 0 0 0 0
    0 0 0 0 0 1 0 0
    0 0 0 0 -1/tau21 -theta21/tau21 0 0
    0 0 0 0 0 0 0 1
    0 0 0 0 0 0 -1/tau22 -theta22/tau22];

Bc=[0 0
    k11/tau11 0
    0 0
    0 k12/tau12
    0 0
    k21/tau21 0
    0 0
```

```
    0 k22/tau22];

C=[1 0 1 0 0 0 0 0;0 0 0 0 1 0 1 0];
D=[0 0;0 0];
plant2=(c2d(ss(Ac,Bc,C,D),Ts));
Pv3=plant2;

[Ap,Bp,C,D]=ssdata(plant2); %state-space matrices
clc
clear all

Models
Ap=A2;Bp=B2;

xs=[3.0423e+000;-3.2205e-015;-4.0423e+000;3.9590e-016;3.2484e+000;1.1481e-015;-
2.2484e+000;-7.0955e-016];
us=[9.3238e-002;1.1928e-001];
%xs=[.5;0;.5;0;-.5;0;-.5;0];

um1 = 1.5-us(1); um2 = 2-us(2);% input constraint

Xm1=um1^2; Xm2=um2^2;


% state weights
Qs1=50*[1 0;0 1]; Qs2=50*[1 0;0 1];
Qs1=C'*Qs1*C+1e-6*eye(8); Qs2=C'*Qs2*C+1e-6*eye(8);
Q1 = 50*eye(2); % state weights
Q1=C'*Q1*C+1e-6*eye(8);
% input weights
R1=1; R2=1;
R = eye(2); % input weight


% no. of sampling time
m=100;

xk = 0*[0;0;0;0;0;0;0;0];
xkhat=xk-xs;
xek=(xk-xs);
yk=C*xkhat;
gdata = zeros(1,m);
tdata = zeros(1,m+1); tdata(1) = 0;
xdata=zeros(8,m+1); xdata(:,1)=xk;
xekdata = zeros(8,m+1); xekdata(:,1) = xek;
FFdata=zeros(2,2,m);
udata=zeros(2,m);
onclock = zeros(1,m);
J=xek'*Qs1*xek;
Jcost=J;
Jnash=xkhat'*Qs1*xkhat;
J3=0;
ydata=zeros(2,m+1); ydata(:,1)=C*xdata(:,1);
ErrTol=1e-2;
MaxIteration=10;
Gdata=zeros(MaxIteration,MaxIteration,m);

F1_old=0*[2.4368e-001  9.6118e-002  2.4342e-001  1.1554e-001 -7.5902e-001 -1.9334e-002 -
7.4396e-001 -1.8754e-001];
F2_old=0*[  8.9819e-001  2.4213e-001  8.9802e-001  2.8776e-001 -3.3902e-001 -1.0614e-002
-3.2376e-001 -1.0193e-001];

x1k_old=xek; x2k_old=xek;
D=[1;0];
dt=1;
alfa=.96;

DD1=[];
DD2=[];
FF=[];
flag=0;
```

```
invQ1=zeros(8,8,25);
invQ2=zeros(8,8,25);
ttime=0;
for k=1:1:m
    tic
   for iterations=1:MaxIteration
       F1_2old=F1_old;
       F2_2old=F2_old;
       [F1,g1,QQ1,YY1]=mpc1fc(A1,A2,B1,B2,R1,Qs1,x1k_old,F2_old,Xm1);
       [F2,g2,QQ2,YY2]=mpc2fc(A1,A2,B1,B2,R2,Qs2,x2k_old,F1_old,Xm2);
%        xk=xkhat;
%         xk'*(A1+B1*[F1;F2])'*inv(QQ1)*(A1+B1*[F1;F2])*xk-xk'*inv(QQ1)*xk
%        xk'*(A2+B2*[F1;F2])'*inv(QQ1)*(A2+B2*[F1;F2])*xk-xk'*inv(QQ1)*xk
%        xk'*(A1+B1*[F1;F2])'*inv(QQ2)*(A1+B1*[F1;F2])*xk-xk'*inv(QQ2)*xk
%        xk'*(A2+B2*[F1;F2])'*inv(QQ2)*(A2+B2*[F1;F2])*xk-xk'*inv(QQ2)*xk
       g1;
       g2;
       flag=1;
       %if k==60
       %    F1
       %    F2
       %    pause
       %end
     %if iterations == MaxIteration
     %      k
     %      F1
     %      F1_old
     %      F2
     %      F2_old
    value=norm([F1;F2]-[F1_old;F2_old]);
    iterations;
    invQ1(1:8,1:8,iterations)=inv(QQ1);
    invQ2(1:8,1:8,iterations)=inv(QQ2);

    %QQ2
    v1(iterations)=QQ1(1,1);
    v2(iterations)=QQ2(1,1);
    %pause
    %   end
    FF=[FF norm([F1;F2])];
   if norm([F1;F2]-[F1_old;F2_old]) <= ErrTol
     %  [F1;F2];
     %  k;iterations; value=norm([F1;F2]-[F1_old;F2_old]);
       break;
   end

   F1_old=alfa*F1+(1-alfa)*F1_2old;
   F2_old=alfa*F2+(1-alfa)*F2_2old;
   DD1=[DD1 g1];
      DD2=[DD2 g2];

   end

   %F1_old=0*F1_2old;
   %F2_old=0*F2_2old;
   disp('========')
   %YY1
   %YY2
   g1;
   g2;
   QQ1;
   QQ2;
   iterations
 toc
ttime=ttime+toc;
   %if iterations == MaxIteration
   %    break;
   %end
   F=[F1;F2]; FFdata(1:2,1:8,k)=F;
   u=F*xkhat
   AC1=A1+B1*F;
```

```
AC2=A2+B2*F;
   setlmis([]);
   P=lmivar(1,[8 1]);

   lmiterm([-1 1 1 P],-AC1',AC1);
   lmiterm([-1 1 1 P],1,1);

   lmiterm([-2 1 1 P],-AC2',AC2);
   lmiterm([-2 1 1 P],1,1);
   lmis3=getlmis;
   options=[0,0,-1,0,1];
   [tmin,xfeas]=feasp(lmis3,options);
   tmin

   if tmin>0
       disp('hohohoho')
       pause;
   end

   xkhat'*inv(.5*QQ1+.5*QQ2)*xkhat
   xkhat=Ap*xkhat+Bp*u; %xkhat=xk-xs;

   xdata(:,k+1)=xkhat;
   ydata(:,k+1)=C*xkhat+[-1;1];% or C*xk
   x1k_old=xkhat; x2k_old=xkhat ;
   tdata(k+1)=k;

end
break
ttime
J=.5*J/m
%J=(1/m)*.5*J
plot(tdata,ydata)
pause
%hold on
udata=[[0;0] udata];
plot(tdata,udata(1,:),tdata,udata(2,:))

ynash=ydata;
unash=udata;
break
save nash ynash unash;



break
if MaxIteration==1
yFC1=ydata;
uFC1=udata;
save FCvenkat1 yFC1 uFC1;
end

if MaxIteration==10
yFC10=ydata;
uFC10=udata;
save FCvenkat10 yFC10 uFC10;
end
if MaxIteration==60
yFC60=ydata;
uFC60=udata;
save FCvenkat60 yFC60 uFC60;
end

%J =
ddd=1;
if ddd==1
    ynash=ydata;
unash=udata;
save venkatnash ynash unash;
end
 % 3.4429e+002
```

```
%J =

%  1.7215e+000

function [F1,g1,QQ,YY]=mpc1fc(A1,A2,Bm1,Bm2,R,Q1,xk,F2,Xm1)
A1=A1+Bm1(:,2)*F2; A2=A2+Bm2(:,2)*F2;

B1=Bm1(:,1); B2=Bm2(:,1);
Q1=Q1+F2'*R*F2;

%Define LMIs
   setlmis([]);
   gama=lmivar(1,[1 0]);
   Q=lmivar(1,[8 1]);
   Y=lmivar(2,[1 8]);
   %Y2=lmivar(2,[1 1]);

   lmiterm([-1 1 1 0],1);
   lmiterm([-1 2 1 0],xk);
   lmiterm([-1 2 2 Q],1 ,1);
%
   %A1=A+Bijv1*F2;
   lmiterm([-2 1 1 Q],1,1);
   lmiterm([-2 2 1 Q],A1,1);
   lmiterm([-2 2 1 Y],B1,1);
   lmiterm([-2 2 2 Q],1,1);
   lmiterm([-2 3 1 Q],real(Q1^0.5),1);
   lmiterm([-2 3 3 gama],1,1);
   lmiterm([-2 4 1 Y],R^0.5,1);
   lmiterm([-2 4 4 gama],1,1);

   %A2=A+Bijv2*F2;
   lmiterm([-3 1 1 Q],1,1);
   lmiterm([-3 2 1 Q],A2,1);
   lmiterm([-3 2 1 Y],B2,1);
   lmiterm([-3 2 2 Q],1,1);
   lmiterm([-3 3 1 Q],real(Q1^0.5),1);
   lmiterm([-3 3 3 gama],1,1);
   lmiterm([-3 4 1 Y],R^0.5,1);
   lmiterm([-3 4 4 gama],1,1);


   %lmiterm([-4 1 1 0],0);
   %lmiterm([-4 2 1 0],Bijv1*F2);
   %lmiterm([-4 2 1 Y2],B1,1);
   %lmiterm([-4 2 2 Q],1,1);
   %lmiterm([-4 3 1 Y2],R^0.5,1);
   %lmiterm([-4 3 3 gama],1,1);


   %lmiterm([-5 1 1 0],0);
   %lmiterm([-5 2 1 0],Bijv2*F2);
   %lmiterm([-5 2 1 Y2],B2,1);
   %lmiterm([-5 2 2 Q],1,1);
   %lmiterm([-5 3 1 Y2],R^0.5,1);
   %lmiterm([-5 3 3 gama],1,1);

   lmiterm([-4 1 1 0],Xm1);
   lmiterm([-4 2 1 -Y],1,1);
   lmiterm([-4 2 2 Q],1 ,1);
%
   %if flag==1
   %    lmiterm([-4 1 1 0],QQ2+1e-1*ones(8,8));
   %    lmiterm([-4 1 1 Q],1,-1);
   %end
    %if flag ==1
    %    lmiterm([-4 1 1 0],Xm2);
    %    lmiterm([-4 2 1 0],-YY2');
    %    lmiterm([-4 2 2 Q],1 ,1);
```

```
    %end

   %lmiterm([-4 1 1 0],1e-1);
   %lmiterm([-4 2 1 Q],1,1);
   %lmiterm([-4 2 1 0],-QQ2);
   %lmiterm([-4 2 2 0],1);
   %end



   LMIs=getlmis;

  c=zeros(1,45); c(1)=1;
  options=[1e-5,100,0,20,1];
   [copt,xopt]=mincx(LMIs,c,options);
   g1=dec2mat(LMIs,xopt,gama); %gdata(1,k)=log(gg);
   QQ=dec2mat(LMIs,xopt,Q);
   YY=dec2mat(LMIs,xopt,Y);
   %YY2=dec2mat(LMIs,xopt,Y2);
   F1=YY*QQ^(-1);
   %F1(2)=YY2*QQ^(-1);
function [F2,g2,QQ,YY]=mpc2fc(A1,A2,Bm1,Bm2,R,Q1,xk,F1,Xm2)
A1=A1+Bm1(:,1)*F1; A2=A2+Bm2(:,1)*F1;

B1=Bm1(:,2); B2=Bm2(:,2);
Q1=Q1+F1'*R*F1;
%Define LMIs
   setlmis([]);
   gama=lmivar(1,[1 0]);
   Q=lmivar(1,[8 1]);
   Y=lmivar(2,[1 8]);
   %Y2=lmivar(2,[1 1]);

   lmiterm([-1 1 1 0],1);
   lmiterm([-1 2 1 0],xk);
   lmiterm([-1 2 2 Q],1 ,1);

   %A1=A+Bijv1*F2;
   lmiterm([-2 1 1 Q],1,1);
   lmiterm([-2 2 1 Q],A1,1);
   lmiterm([-2 2 1 Y],B1,1);
   lmiterm([-2 2 2 Q],1,1);
   lmiterm([-2 3 1 Q],real(Q1^0.5),1);
   lmiterm([-2 3 3 gama],1,1);
   lmiterm([-2 4 1 Y],R^0.5,1);
   lmiterm([-2 4 4 gama],1,1);

   %A2=A+Bijv2*F2;
   lmiterm([-3 1 1 Q],1,1);
   lmiterm([-3 2 1 Q],A2,1);
   lmiterm([-3 2 1 Y],B2,1);
   lmiterm([-3 2 2 Q],1,1);
   lmiterm([-3 3 1 Q],real(Q1^0.5),1);
   lmiterm([-3 3 3 gama],1,1);
   lmiterm([-3 4 1 Y],R^0.5,1);
   lmiterm([-3 4 4 gama],1,1);


   %lmiterm([-4 1 1 0],0);
   %lmiterm([-4 2 1 0],Bijv1*F2);
   %lmiterm([-4 2 1 Y2],B1,1);
   %lmiterm([-4 2 2 Q],1,1);
   %lmiterm([-4 3 1 Y2],R^0.5,1);
   %lmiterm([-4 3 3 gama],1,1);


   %lmiterm([-5 1 1 0],0);
   %lmiterm([-5 2 1 0],Bijv2*F2);
   %lmiterm([-5 2 1 Y2],B2,1);
   %lmiterm([-5 2 2 Q],1,1);
   %lmiterm([-5 3 1 Y2],R^0.5,1);
```

```
   %lmiterm([-5 3 3 gama],1,1);
%
   lmiterm([-4 1 1 0],Xm2);
   lmiterm([-4 2 1 -Y],1,1);
   lmiterm([-4 2 2 Q],1 ,1);
%
   %if flag==1
   %    lmiterm([-4 1 1 0],QQ1+1e-1*ones(8,8));
   %    lmiterm([-4 1 1 Q],1,-1);
   %end
    %if flag ==1
    %    lmiterm([-4 1 1 0],Xm1);
    %    lmiterm([-4 2 1 0],-YY1');
    %    lmiterm([-4 2 2 Q],1 ,1);
    %end

   %lmiterm([-4 1 1 0],1e-1);
   %lmiterm([-4 2 1 Q],1,1);
   %lmiterm([-4 2 1 0],-QQ1);
   %lmiterm([-4 2 2 0],1);
   %end
   LMIs=getlmis;

  c=zeros(1,45); c(1)=1;
 options=[1e-5,100,0,20,1];
   [copt,xopt]=mincx(LMIs,c,options);
   g2=dec2mat(LMIs,xopt,gama); %gdata(1,k)=log(gg);
   QQ=dec2mat(LMIs,xopt,Q);
   YY=dec2mat(LMIs,xopt,Y);
   %YY2=dec2mat(LMIs,xopt,Y2);
   F2=YY*QQ^(-1);
   %F1(2)=YY2*QQ^(-1);
clc
clear all

g11=tf(4.05,[50 1],'IODELAY',6);
g12=tf(1.77,[60 1],'IODELAY',6);
g13=tf(5.88,[50 1],'IODELAY',6);

g21=tf(5.39,[50 1],'IODELAY',4);
g22=tf(5.72,[60 1],'IODELAY',2);
g23=tf(6.9,[40 1],'IODELAY',2);

g31=tf(4.38,[33 1],'IODELAY',4);
g32=tf(4.42,[44 1],'IODELAY',4);
g33=tf(7.2,[19 1]);


Plant=[g11 g12 g13;g21 g22 g23;g31 g32 g33]; % Transfer Matrix in Continous
Ts=2;

% Plantd=c2d(Plant,Ts);
%
% step(Plant,Plantd)

%y1
% Transfer function:
%            0.1588
% z^(-3) * ----------
%          z - 0.9608

A11=[.9608 1 0 0;0 0 1 0;0 0 0 1;0 0 0 0];
B11=[0;0;0;.1588];
C11=[1 0 0 0];
D11=0;

% Transfer function:
%           0.05803
% z^(-3) * ----------
%          z - 0.9672
%
```

```
% Sampling time: 2
A12=[.9672 1 0 0;0 0 1 0;0 0 0 1;0 0 0 0];
B12=[0;0;0;.05803];
C12=[1 0 0 0];
D12=0;

% Transfer function:
%            0.2306
% z^(-3) * ----------
%          z - 0.9608
%
% Sampling time: 2

A13=[.9608 1 0 0;0 0 1 0;0 0 0 1;0 0 0 0];
B13=[0;0;0;.2306];
C13=[1 0 0 0];
D13=0;

A1=blkdiag(A11,A12,A13);
B1=blkdiag(B11,B12,B13);
C1=[C11 C12 C13];
D1=zeros(1,3);
% model=[g11 g12 g13];
% step(model,ss(A1,B1,C1,D1,Ts))
```

Key MATLAB codes used in Chapter 5

```
clc
clear
Ts=4;
Ac=[-1/75 0;0 -1/75];
dt=1;
Bc1=[.878/75 .864*dt/75;1.082*dt/75 1.096/75]*[1.8 0;0 0.2];
Bc2=[.878/75 .864*dt/75;1.082*dt/75 1.096/75]*[1. 0;0 1.];
B3=[.878/75 .864/75;1.082/75 1.096/75]*[1.5 0;0 0.5];
%B4=[.878/75 .864/75;1.082/75 1.096/75]*[0.8 0;0 1.2];
[A,B1,C,D]=ssdata(c2d(ss(Ac,Bc1,eye(2),zeros(2)),Ts));
[A,B2,C,D]=ssdata(c2d(ss(Ac,Bc2,eye(2),zeros(2)),Ts));

nc=6;
nu=2;
nu1=1;
nu2=1;
nx=2;

K=[-7.1412e-002  9.5463e-003;-6.9765e-002  9.4318e-003];
K =[-1.2784e-001 -1.5752e-001;-5.5039e-002 -6.9872e-002];

M1=[zeros((nc-1)*nu1,nu1) eye((nc-1)*nu1);zeros(nu1,nu1) zeros(nu1,(nc-1)*nu1)];

M2=[zeros((nc-1)*nu1,nu2) eye((nc-1)*nu2);zeros(nu1,nu2) zeros(nu1,(nc-1)*nu2)];

T=[eye(nu) zeros(nu,nu*nc-2)];
T1=[eye(nu1) zeros(nu1,nu*nc-1)]
T2=[zeros(nu2,nu2*nc) eye(nu2) zeros(nu2,nu2*nc-1)]

epsi1=[(A+B1*K) [B1(:,1) zeros(nx,(nc-1)*nu1)] [B1(:,2) zeros(nx,(nc-
1)*nu2)];zeros(nu1*nc,nx) M1 0*M1;zeros(nu2*nc,nx) 0*M2 M2];
epsi2=[(A+B2*K) [B2(:,1) zeros(nx,(nc-1)*nu1)] [B2(:,2) zeros(nx,(nc-
1)*nu2)];zeros(nu1*nc,nx) M1 0*M1;zeros(nu2*nc,nx) 0*M2 M2];

Y = sdpvar(nu*nc+nx,nu*nc+nx);

F = set( [Y Y*epsi1';epsi1*Y Y] >0);
F = F + set( [Y Y*epsi2';epsi2*Y Y] >0);
F = F + set([K(1,:) T1]*Y*[K(1,:) T1]' <1.0000e-002);
F = F + set([K(2,:) T2]*Y*[K(2,:) T2]' <1.0000e-002);
```

```
%ops = sdpsettings('verbose',1,'sedumi.eps',1e-4);
ops = sdpsettings('verbose',1,'sedumi.bigeps',1e-2);

solution = solvesdp(F,-geomean([eye(nx) zeros(nx,nc*nu)]*Y*[eye(nx)
zeros(nx,nc*nu)]'),ops);
any_probelms=solution.problem
Y = double(Y);

Tx=[eye(nx) zeros(nx,nu*nc)];

Tf=[zeros(nx,nx+nu*nc);zeros(nu*nc,nx) eye(nu*nc)];
Qinv=Y
Qxinv=Tx*(Y)*Tx'
Qfinv1=Y(nx+1:nx+nc*nu1,nx+1:nx+nc*nu1)%Tf*inv(Y)*Tf'
Qfinv2=Y(nx+nc*nu1+1:nx+nc*nu1+nc*nu2,nx+nc*nu1+1:nx+nc*nu1+nc*nu2)

Qf1x=Y(nx+1:nx+nc*nu1,1:nx)%Tf*inv(Y)*Tx'
Qf2x=Y(nx+nc*nu1+1:end,1:nx)

Qf1f2=Y(nx+nc*nu2+1:end,nx+1:nx+nc*nu1)

%plant
B3=[.878/75 .864/75;1.082/75 1.096/75]*[1.5 0;0 0.5];
[Ap,Bp,C,D]=ssdata(c2d(ss(Ac,B3,eye(2),zeros(2)),Ts));


C = [1 0;0 1];

m=250;
xk = [-1;0];
udata=zeros(2,m);
tdata = zeros(1,m+1); tdata(1) = 0;
xdata=zeros(2,m+1); xdata(:,1)=xk+[1;0];

ErrTol=1e-3;
MaxIteration=1000;
f1_old=-.001*ones(1,nc*nu1);
f2_old=-.001*ones(1,nc*nu2);
fc =[-4.3735e-002 -1.8869e-002 -3.2630e-002 -1.3935e-002 -2.3657e-002 -9.9938e-003 -
1.6301e-002 -6.8079e-003 -1.0135e-002 -4.1831e-003 -4.8024e-003 -1.9581e-003];
fc=[-4.3833e-002 -3.2724e-002 -2.3741e-002 -1.6370e-002 -1.0187e-002 -4.8341e-003 -
1.8919e-002 -1.3985e-002 -1.0039e-002 -6.8446e-003 -4.2100e-003 -1.9742e-003];
%   f1_old=[fc(1) fc(3) fc(5) fc(7) fc(9) fc(11)];
%   f2_old=[fc(2) fc(4) fc(6) fc(8) fc(10) fc(12)];
f1_old=0*fc(1:nu1*nc);
f2_old=0*fc(nu1*nc+1:nu*nc);

x1k_old=xk; x2k_old=xk;
ru=1e1;
tic
Lam1=[];
Lam2=[];
for k=1:1:1
    tic
   for iterations=1:MaxIteration

[f1,alfa1,lamda1]=DMPC1(ru,nu1,nc,Qxinv,Qfinv1,Qfinv2,Qf1x,Qf2x,Qf1f2,x1k_old,f2_old);

[f2,alfa2,lamda2]=DMPC2(ru,nu2,nc,Qxinv,Qfinv1,Qfinv2,Qf1x,Qf2x,Qf1f2,x2k_old,f1_old);



   if norm(f1-f1_old) <= ErrTol && norm(f2-f2_old)<=ErrTol
          break;
   end
   lamda1
   lamda2
   f1_old=f1;
   f2_old=f2;
   Lam1=[Lam1 lamda1];
   Lam2=[Lam2 lamda2];
```

```
   end
c1=[xk' f1 f2_old]*inv(Qinv)*[xk;f1';f2_old']
c2=[xk' f1_old f2]*inv(Qinv)*[xk;f1_old';f2']

   %F1_old=0*F1_2old;
   %F2_old=0*F2_2old;
   disp('========')
   %YY1
   %YY2
   alfa1;
   alfa2'
   f1;
   f2;
   toc
   iterations


   %if iterations == MaxIteration
   %    break;
   %end
   F=[f1(1);f2(1)]; %FFdata(1:2,1:8,k)=F;
   u=K*xk+F; udata(1:2,k)=u;

%     Ap = Ap + DA;
%       Bp = Bp + DB;
 xk=Ap*xk+Bp*u;
 if k==150
     xk=xk+[.5;.2];
 end
 if k==151
     pause;
 end
           xdata(:,k+1)=xk+[1;0];
    x1k_old=xk; x2k_old=xk;

   tdata(k+1) = k;

end
toc
x=1:12;
fm=[f1 f2];
plot(x(1:6),fc(1:6),x(7:12),fc(7:12),x,fm,':')

xdmpc=xdata;
udmpc=udata;
break
save HP_DMPC xdmpc udmpc
plot(tdata,xdata)
pause
%hold on
plot(tdata(2:end),udata(1,:),tdata(2:end),udata(2,:))

function
[f1,alfa1,lamda]=DMPC1(ru,nu1,nc,Qxinv,Qfinv1,Qfinv2,Qf1x,Qf2x,Qf1f2,x1k_old,f2_old);

 setlmis([]);
    a=lmivar(1,[1 0]);
    lambda=lmivar(1,[1 0]);
    ff=lmivar(2,[1 nu1*nc]);


    lmiterm([-1 1 1 a],1,1);
    %lmiterm([-1 1 1 lambda],1,ru);
    lmiterm([-1 1 2 ff],1,1);
    lmiterm([-1 1 3 0],f2_old);
    lmiterm([-1 1 4 lambda],1,1);
    lmiterm([-1 2 2 0],1);
    lmiterm([-1 3 3 0],1);
    lmiterm([-1 4 4 0],ru^(-1));
```

```
    lmiterm([-2 1 1 lambda],1,1);
    lmiterm([-2 1 1 0],1);
    lmiterm([-2 1 2 0],x1k_old');
    lmiterm([-2 1 3 ff],1,1);
    lmiterm([-2 1 4 0],f2_old);
    lmiterm([-2 2 2 0],Qxinv);
    lmiterm([-2 2 3 0],Qf1x');
    lmiterm([-2 2 4 0],Qf2x');
    lmiterm([-2 3 3 0],Qfinv1);
    lmiterm([-2 3 4 0],Qf1f2');
    lmiterm([-2 4 4 0],Qfinv2);

     LMIs=getlmis;

  c=zeros(1,2+nc*nu1); c(1)=1;
  options=[1e-3,1000,1e9,100,1];
   [copt,xopt]=mincx(LMIs,c,options);
   alfa1=dec2mat(LMIs,xopt,a);
   f1=dec2mat(LMIs,xopt,ff);
    lamda=dec2mat(LMIs,xopt,lambda);
function
[f2,alfa2,lamda]=DMPC2(ru,nu2,nc,Qxinv,Qfinv1,Qfinv2,Qf1x,Qf2x,Qf1f2,x2k_old,f1_old);

 setlmis([]);
    a=lmivar(1,[1 0]);
    lambda=lmivar(1,[1 0]);
    ff=lmivar(2,[1 nu2*nc]);


    lmiterm([-1 1 1 a],1,1);
    %lmiterm([-1 1 1 lambda],1,ru);
    lmiterm([-1 1 2 0],f1_old);
    lmiterm([-1 1 3 ff],1,1);
    lmiterm([-1 1 4 lambda],1,1);
    lmiterm([-1 2 2 0],1);
    lmiterm([-1 3 3 0],1);
    lmiterm([-1 4 4 0],ru^(-1));

    lmiterm([-2 1 1 lambda],1,1);
    lmiterm([-2 1 1 0],1);
    lmiterm([-2 1 2 0],x2k_old');
    lmiterm([-2 1 3 0],f1_old);
    lmiterm([-2 1 4 ff],1,1);
    lmiterm([-2 2 2 0],Qxinv);
    lmiterm([-2 2 3 0],Qf1x');
    lmiterm([-2 2 4 0],Qf2x');
    lmiterm([-2 3 3 0],Qfinv1);
    lmiterm([-2 3 4 0],Qf1f2');
    lmiterm([-2 4 4 0],Qfinv2);

     LMIs=getlmis;

  c=zeros(1,2+nc*nu2); c(1)=1;
  options=[1e-3,1000,1e9,100,1];
   [copt,xopt]=mincx(LMIs,c,options);
   alfa2=dec2mat(LMIs,xopt,a);
   f2=dec2mat(LMIs,xopt,ff);
    lamda=dec2mat(LMIs,xopt,lambda);


clc
clear
model1
Am1=A;Bm1=B;Cm1=C;Dm1=D;

model2
Am2=A;Bm2=B;Cm2=C;Dm2=D;

%nominal

An=Am1;Bn=.5*Bm1+.5*Bm2; Cn=Cm1;
```

```
steady_states=inv([eye(21)-An -Bn;Cn zeros(3,3)])*[zeros(21,1);3;3;-3];
xs=steady_states(1:21);
us=steady_states(22:end);

nc=12;
nu=3;
nx=21;
nu1=1;
nu2=1;
nu3=1;

M1=[zeros((nc-1)*nu1,nu1) eye((nc-1)*nu1);zeros(nu1,nu1) zeros(nu1,(nc-1)*nu1)];

M2=[zeros((nc-1)*nu2,nu2) eye((nc-1)*nu2);zeros(nu2,nu2) zeros(nu2,(nc-1)*nu2)];

M3=[zeros((nc-1)*nu3,nu3) eye((nc-1)*nu3);zeros(nu3,nu3) zeros(nu3,(nc-1)*nu3)];

load uncgain

%  K=[-2.2442e-001 -2.1832e-001 -2.0040e-001 -1.7054e-001 -2.2761e-001 -2.2017e-001 -
2.0099e-001 -1.7003e-001 -2.2442e-001 -2.1832e-001 -2.0039e-001 -1.7054e-001 -1.9227e-001
-1.7914e-001 -1.9912e-001 -1.8324e-001  1.1160e-001  8.3292e-002  1.1439e-001  8.4962e-
002  1.0067e-001
%    1.6290e-001  1.6908e-001  1.7538e-001  1.8243e-001  1.9021e-001  1.9619e-001
2.0225e-001  2.0901e-001  1.6290e-001  1.6908e-001 1.7538e-001  1.8243e-001 -4.0610e-001
-4.2027e-001 -4.2428e-001 -3.8187e-001 -5.9984e-002 -6.5663e-002 -7.2452e-002 -7.7721e-
002 -4.0862e-002
%   -2.3734e-001 -2.4937e-001 -2.6257e-001 -2.7630e-001 -2.5710e-001 -2.6815e-001 -
2.8025e-001 -2.9275e-001 -2.3734e-001 -2.4937e-001 -2.6257e-001 -2.7630e-001 -2.5098e-001
-2.6977e-001 -2.5532e-001 -2.4466e-001 -5.4042e-001 -5.6642e-001 -5.4574e-001 -5.6345e-
001 -5.1078e-001];


T=[eye(nu) zeros(nu,nu*nc-2)];
T1=[eye(nu1) zeros(nu1,nu1*(nc-1)) zeros(nu1,(nu-nu1)*nc)];
T2=[zeros(nu2,nu2*nc) eye(nu2) zeros(nu2,nu2*(nc-1)) zeros(nu2,nu2*nc)];
T3=[zeros(nu3,(nu-nu3)*nc) eye(nu3) zeros(nu3,nu3*(nc-1))];

epsi1=[(Am1+Bm1*K) [Bm1(:,1) zeros(nx,(nc-1)*nu1)] [Bm1(:,2) zeros(nx,(nc-1)*nu2)]
[Bm1(:,3) zeros(nx,(nc-1)*nu3)];zeros(nu1*nc,nx) M1 0*M1 0*M1;zeros(nu2*nc,nx) 0*M2 M2
0*M2;zeros(nu3*nc,nx) 0*M3 0*M3 M3];
epsi2=[(Am2+Bm2*K) [Bm2(:,1) zeros(nx,(nc-1)*nu1)] [Bm2(:,2) zeros(nx,(nc-1)*nu2)]
[Bm2(:,3) zeros(nx,(nc-1)*nu3)];zeros(nu1*nc,nx) M1 0*M1 0*M1;zeros(nu2*nc,nx) 0*M2 M2
0*M2;zeros(nu3*nc,nx) 0*M3 0*M3 M3];

% Y = sdpvar(nu*nc+nx,nu*nc+nx);
%
% F = set( [Y Y*epsi1';epsi1*Y Y] >0);
% F = F + set( [Y Y*epsi2';epsi2*Y Y] >0);
% F = F + set([K(1,:) T1]*Y*[K(1,:) T1]' <(10-us(1))^2);
% F = F + set([K(2,:) T2]*Y*[K(2,:) T2]' <(10-us(2))^2);
% F = F + set([K(3,:) T3]*Y*[K(3,:) T3]' <(10-us(3))^2);
%
% %ops = sdpsettings('verbose',1,'sedumi.eps',1e-4);
% ops = sdpsettings('verbose',1,'solver','sdpt3');
%
% solution = solvesdp(F,-geomean([eye(nx) zeros(nx,nc*nu)]*Y*[eye(nx)
zeros(nx,nc*nu)]'),ops),sdpsettings('debug',1)
% any_probelms=solution.problem
% Y = double(Y);

load ShahY;


Tx=[eye(nx) zeros(nx,nu*nc)];

Tf=[zeros(nx,nx+nu*nc);zeros(nu*nc,nx) eye(nu*nc)];
Qinv=Y;
Qxinv=Tx*(Y)*Tx';
Qfinv1=Y(nx+1:nx+nc*nu1,nx+1:nx+nc*nu1);%Tf*inv(Y)*Tf'
```

```
Qfinv2=Y(nx+nc*nu1+1:nx+nc*nu1+nc*nu2,nx+nc*nu1+1:nx+nc*nu1+nc*nu2);
Qfinv3=Y(nx+nc*nu1+nc*nu2+1:nx+nc*nu1+nc*nu2+nc*nu3,nx+nc*nu1+nc*nu2+1:nx+nc*nu1+nc*nu2+n
c*nu3);

Qf1x=Y(nx+1:nx+nc*nu1,1:nx);%Tf*inv(Y)*Tx'
Qf2x=Y(nx+nc*nu1+1:nx+nc*nu1+nc*nu2,1:nx);
Qf3x=Y(nx+nc*nu1++nc*nu2+1:end,1:nx);

Qf1f2=Y(nx+nc*nu1+1:nx+nc*nu1+nc*nu2,nx+1:nx+nc*nu1);
Qf1f3=Y(nx+nc*nu1+nc*nu2+1:end,nx+1:nx+nc*nu1);
Qf2f3=Y(nx+nc*nu1+nc*nu2+1:end,nx+nc*nu1+1:nx+nc*nu1+nc*nu2);

xk = 0*xs;
xkhat=(xk-xs);
Ap=Am1;Bp=Bm1;
m=160;
% Ap=A2;
% Bp=B2;
% DA=(A1-A2)/m;
% DB = (B1-B2)/m;
udata=zeros(3,m);
tdata = zeros(1,m+1); tdata(1) = 0;
xdata=zeros(nx,m+1); xdata(:,1)=xk;
ydata=zeros(3,m+1); ydata(:,1)=Cn*xdata(:,1);
ErrTol=1e-2;
MaxIteration=40;
fc =[-1.2156e+000 -8.1986e-001 -5.7332e-001 -4.0091e-001 -2.7185e-001 -1.7290e-001 -
9.6617e-002 -3.8178e-002  5.8883e-003 3.8256e-002  6.1060e-002  7.6027e-002  8.4565e-002
8.7809e-002  8.6668e-002  8.1842e-002  7.3824e-002  6.2857e-002 4.9195e-002  3.0734e-002
-1.9962e-003  4.8688e-003  6.6055e-003  6.1084e-003  4.6971e-003  2.9850e-003  1.2745e-
003 -2.9015e-004 -1.6491e-003 -2.7902e-003 -3.7298e-003 -4.5023e-003 -5.1561e-003 -
5.7534e-003 -6.3780e-003 -7.1520e-003 -8.2657e-003 -1.0058e-002 -1.2438e-002 -1.6468e-002
1.1545e-001  8.4060e-002  5.5057e-002  3.0641e-002  1.1146e-002 -3.8538e-003 -1.4980e-002
-2.2879e-002 -2.8145e-002 -3.1302e-002 -3.2797e-002 -3.3009e-002 -3.2268e-002 -3.0886e-
002 -2.9204e-002 -2.7685e-002 -2.7065e-002 -2.8650e-002 -3.4580e-002 -5.0834e-002];
%fc = [-2.4038e+000 -1.7873e+000 -1.3787e+000 -1.0769e+000 -8.3846e-001 -6.4483e-001 -
4.8582e-001 -3.5507e-001 -2.4810e-001 -1.6148e-001 -9.2400e-002 -3.8511e-002  2.1954e-003
3.1443e-002  5.0702e-002  6.1202e-002  6.3929e-002  5.9607e-002 4.9206e-002  3.1150e-002
-1.6853e-001 -8.6869e-002 -3.9279e-002 -1.0403e-002  7.3347e-003  1.7996e-002  2.3959e-
002 2.6693e-002  2.7147e-002  2.5948e-002  2.3511e-002  2.0105e-002  1.5887e-002
1.0928e-002  5.2189e-003 -1.3273e-003 -8.8575e-003 -1.7617e-002 -2.5176e-002 -3.1807e-002
1.9129e-001  1.8864e-001  1.5509e-001  1.1395e-001  7.4488e-002 4.0036e-002  1.1489e-002
-1.1309e-002 -2.8946e-002 -4.2156e-002 -5.1691e-002 -5.8281e-002 -6.2647e-002 -6.5546e-
002 -6.7874e-002 -7.0829e-002 -7.6070e-002 -8.5806e-002 -1.0080e-001 -1.2517e-001];
fc=[-1.2668e+000 -9.1538e-001 -7.0035e-001 -5.4837e-001 -4.3056e-001 -3.3551e-001 -
2.5738e-001 -1.9280e-001 -1.3947e-001 -9.5671e-002 -5.9885e-002 -3.0231e-002  2.1346e-002
2.1039e-002  1.8703e-002  1.5845e-002  1.3065e-002  1.0567e-002 8.4120e-003  6.6102e-003
5.1753e-003  4.1714e-003  3.8768e-003  4.7859e-003  1.4762e-001  1.2251e-001  9.6519e-002
7.2656e-002  5.1888e-002  3.4266e-002  1.9594e-002  7.6414e-003 -1.7349e-003 -8.4336e-003
-1.1683e-002 -9.3704e-003];
f1_old=zeros(1,nc*nu1);
f2_old=zeros(1,nc*nu2);
f3_old=zeros(1,nc*nu3);

f1_old=.2*randn(1,nc*nu1);
f2_old=.2*randn(1,nc*nu2);
f3_old=.2*randn(1,nc*nu3);


%
% f1_old=fc(1:nu1*nc);
% f2_old=fc(nu1*nc+1:nu1*nc+nu2*nc);
% f3_old=fc(nu1*nc+nu2*nc+1:end);

x1k_old=xkhat; x2k_old=xkhat;x3k_old=xkhat;
Lam1=[];
Lam2=[];
Lam3=[];


ru=1e2;
```

```
fg=.95;
terminator=1;
for k=1:1:1
    tic
   for iterations=1:MaxIteration
       f1_2old=f1_old;
       f2_2old=f2_old;
       f3_2old=f3_old;

[f1,alfa1,lamda1]=DMPC1(terminator,ru,nu1,nc,Qxinv,Qfinv1,Qfinv2,Qfinv3,Qf1x,Qf2x,Qf3x,Qf
1f2,Qf1f3,Qf2f3,x1k_old,f2_old,f3_old);

[f2,alfa2,lamda2]=DMPC2(terminator,ru,nu2,nc,Qxinv,Qfinv1,Qfinv2,Qfinv3,Qf1x,Qf2x,Qf3x,Qf
1f2,Qf1f3,Qf2f3,x2k_old,f1_old,f3_old);

[f3,alfa3,lamda3]=DMPC3(terminator,ru,nu3,nc,Qxinv,Qfinv1,Qfinv2,Qfinv3,Qf1x,Qf2x,Qf3x,Qf
1f2,Qf1f3,Qf2f3,x3k_old,f1_old,f2_old);

   % terminator=exp(-.5*iterations);
c1=[xkhat' f1 f2_old f3_old]*inv(Qinv)*[xkhat;f1';f2_old';f3_old']
c2=[xkhat' f1_old f2 f3_old]*inv(Qinv)*[xkhat;f1_old';f2';f3_old']
c3=[xkhat' f1_old f2_old f3]*inv(Qinv)*[xkhat;f1_old';f2_old';f3']
   Lam1=[Lam1 lamda1];
   Lam2=[Lam2 lamda2];
   Lam3=[Lam3 lamda3];

   if norm(f1-f1_old) <= ErrTol&&norm(f2-f2_old)<=ErrTol&&norm(f3-f3_old)<=ErrTol
         break;
   end
%    lamda1
% %    lamda2
%     f1_old=f1
%     f2_old=f2
%     f3_old=f3

   f1_old=fg*f1+(1-fg)*f1_2old;
   f2_old=fg*f2+(1-fg)*f2_2old;
   f3_old=fg*f3+(1-fg)*f3_2old;


   end

   %F1_old=0*F1_2old;
   %F2_old=0*F2_2old;
   disp('========')
   %YY1
   %YY2


   iterations
 toc

   %if iterations == MaxIteration
   %    break;
   %end
   F=[f1(1);f2(1);f3(1)]; %FFdata(1:2,1:8,k)=F;
   u=K*xkhat+F; udata(1:3,k)=u;

%     Ap = Ap + DA;
%       Bp = Bp + DB;
  xkhat=Ap*xkhat+Bp*u %xkhat=xk-xs;
    yk=C*xkhat;
    x1k_old=xkhat; x2k_old=xkhat;x3k_old=xkhat;
      xdata(:,k+1)=xkhat;
      ydata(:,k+1)=C*xkhat+[3;3;-3];

   tdata(k+1) = k;

end
x=1:nc*nu;
fm=[f1 f2 f3];
```

```
plot(x,fc,x,fm,':')
ydmpc=ydata;
udmpc=udata;
break
save Shah_DMPC ydmpc udmpc


function
[f1,alfa1,lamda]=DMPC1(t,ru,nu1,nc,Qxinv,Qfinv1,Qfinv2,Qfinv3,Qf1x,Qf2x,Qf3x,Qf1f2,Qf1f3,
Qf2f3,x1k_old,f2_old,f3_old);

 setlmis([]);
    a=lmivar(1,[1 0]);
    lambda=lmivar(1,[1 0]);
    ff=lmivar(2,[1 nu1*nc]);

    lmiterm([-1 1 1 a],1,1);
    %lmiterm([-1 1 1 lambda],-ru,1);
    lmiterm([-1 1 2 ff],1,1);
    lmiterm([-1 1 3 0],f2_old);
    lmiterm([-1 1 4 0],f3_old);
    lmiterm([-1 1 5 lambda],t,1);
    lmiterm([-1 2 2 0],1);
    lmiterm([-1 3 3 0],1);
    lmiterm([-1 4 4 0],1);
    lmiterm([-1 5 5 0],ru^(-1));

    lmiterm([-2 1 1 lambda],t,1);
    lmiterm([-2 1 1 0],1);
    lmiterm([-2 1 2 0],x1k_old');
    lmiterm([-2 1 3 ff],1,1);
    lmiterm([-2 1 4 0],f2_old);
    lmiterm([-2 1 5 0],f3_old);
    lmiterm([-2 2 2 0],Qxinv);
    lmiterm([-2 2 3 0],Qf1x');
    lmiterm([-2 2 4 0],Qf2x');
    lmiterm([-2 2 5 0],Qf3x');
    lmiterm([-2 3 3 0],Qfinv1);
    lmiterm([-2 3 4 0],Qf1f2');
    lmiterm([-2 3 5 0],Qf1f3');
    lmiterm([-2 4 4 0],Qfinv2);
    lmiterm([-2 4 5 0],Qf2f3');
    lmiterm([-2 5 5 0],Qfinv3');

     LMIs=getlmis;
  %ndec = decnbr(LMIs);
  c=zeros(1,2+nc*nu1); c(1)=1;
  options=[1e-2,1000,1e9,100,1];
   [copt,xopt]=mincx(LMIs,c,options);


    alfa1=dec2mat(LMIs,xopt,a);
   f1=dec2mat(LMIs,xopt,ff);
   lamda=dec2mat(LMIs,xopt,lambda);

%c1=x1k_old'*Qxinv*x1k_old+2*x1k_old'*Qf1x'*f1'+2*x1k_old'*Qf2x'*f2_old'+2*x1k_old'*Qf3x'
*f3_old'+f1*Qfinv1*f1'+2*f1*Qf1f2*f2_old'+2*f1*Qf1f3*f3_old'+f2_old*Qfinv2*f2_old'+2*f2_o
ld*Qf2f3*f3_old'+f3_old*Qfinv3*f3_old';




function
[f2,alfa2,lamda]=DMPC2(t,ru,nu2,nc,Qxinv,Qfinv1,Qfinv2,Qfinv3,Qf1x,Qf2x,Qf3x,Qf1f2,Qf1f3,
Qf2f3,x2k_old,f1_old,f3_old);

 setlmis([]);
    a=lmivar(1,[1 0]);
    lambda=lmivar(1,[1 0]);
    ff=lmivar(2,[1 nu2*nc]);
```

```
    lmiterm([-1 1 1 a],1,1);
    %lmiterm([-1 1 1 lambda],-ru,1);
    lmiterm([-1 1 2 0],f1_old);
    lmiterm([-1 1 3 ff],1,1);
    lmiterm([-1 1 4 0],f3_old);
    lmiterm([-1 1 5 lambda],t,1);
    lmiterm([-1 2 2 0],1);
    lmiterm([-1 3 3 0],1);
    lmiterm([-1 4 4 0],1);
    lmiterm([-1 5 5 0],ru^(-1));

    lmiterm([-2 1 1 lambda],t,1);
    lmiterm([-2 1 1 0],1);
    lmiterm([-2 1 2 0],x2k_old');
    lmiterm([-2 1 3 0],f1_old);
    lmiterm([-2 1 4 ff],1,1);
    lmiterm([-2 1 5 0],f3_old);
    lmiterm([-2 2 2 0],Qxinv);
    lmiterm([-2 2 3 0],Qf1x');
    lmiterm([-2 2 4 0],Qf2x');
    lmiterm([-2 2 5 0],Qf3x');
    lmiterm([-2 3 3 0],Qfinv1);
    lmiterm([-2 3 4 0],Qf1f2');
    lmiterm([-2 3 5 0],Qf1f3');
    lmiterm([-2 4 4 0],Qfinv2);
    lmiterm([-2 4 5 0],Qf2f3');
    lmiterm([-2 5 5 0],Qfinv3');

     LMIs=getlmis;

  c=zeros(1,2+nc*nu2); c(1)=1;
  options=[1e-2,1000,1e9,100,1];
   [copt,xopt]=mincx(LMIs,c,options);
   alfa2=dec2mat(LMIs,xopt,a);
   f2=dec2mat(LMIs,xopt,ff);
   lamda=dec2mat(LMIs,xopt,lambda);

%c2=x2k_old'*Qxinv*x2k_old+2*x2k_old'*Qf1x'*f1_old'+2*x2k_old'*Qf2x'*f2'+2*x2k_old'*Qf3x'
*f3_old'+f1_old*Qfinv1*f1_old'+2*f1_old*Qf1f2*f2'+2*f1_old*Qf1f3*f3_old'+f2*Qfinv2*f2'+2*
f2*Qf2f3*f3_old'+f3_old*Qfinv3*f3_old';
function
[f3,alfa3,lamda]=DMPC3(t,ru,nu3,nc,Qxinv,Qfinv1,Qfinv2,Qfinv3,Qf1x,Qf2x,Qf3x,Qf1f2,Qf1f3,
Qf2f3,x3k_old,f1_old,f2_old);

 setlmis([]);
    a=lmivar(1,[1 0]);
    lambda=lmivar(1,[1 0]);
    ff=lmivar(2,[1 nu3*nc]);

    lmiterm([-1 1 1 a],1,1);
    %lmiterm([-1 1 1 lambda],-ru,1);
    lmiterm([-1 1 2 0],f1_old);
    lmiterm([-1 1 3 0],f2_old);
    lmiterm([-1 1 4 ff],1,1);
    lmiterm([-1 1 5 lambda],t,1);
    lmiterm([-1 2 2 0],1);
    lmiterm([-1 3 3 0],1);
    lmiterm([-1 4 4 0],1);
    lmiterm([-1 5 5 0],ru^(-1));

    lmiterm([-2 1 1 lambda],t,1);
    lmiterm([-2 1 1 0],1);
    lmiterm([-2 1 2 0],x3k_old');
    lmiterm([-2 1 3 0],f1_old);
    lmiterm([-2 1 4 0],f2_old);
    lmiterm([-2 1 5 ff],1,1);
    lmiterm([-2 2 2 0],Qxinv);
    lmiterm([-2 2 3 0],Qf1x');
    lmiterm([-2 2 4 0],Qf2x');
    lmiterm([-2 2 5 0],Qf3x');
    lmiterm([-2 3 3 0],Qfinv1);
```

```
    lmiterm([-2 3 4 0],Qf1f2');
    lmiterm([-2 3 5 0],Qf1f3');
    lmiterm([-2 4 4 0],Qfinv2);
    lmiterm([-2 4 5 0],Qf2f3');
    lmiterm([-2 5 5 0],Qfinv3');

     LMIs=getlmis;

  c=zeros(1,2+nc*nu3); c(1)=1;
  options=[1e-2,1000,1e9,100,1];
   [copt,xopt]=mincx(LMIs,c,options);
   alfa3=dec2mat(LMIs,xopt,a);
   f3=dec2mat(LMIs,xopt,ff);
   lamda=dec2mat(LMIs,xopt,lambda);

%c3=x3k_old'*Qxinv*x3k_old+2*x3k_old'*Qf1x'*f1_old'+2*x3k_old'*Qf2x'*f2_old'+2*x3k_old'*Q
f3x'*f3'+f1_old*Qfinv1*f1_old'+2*f1_old*Qf1f2*f2_old'+2*f1_old*Qf1f3*f3'+f2_old*Qfinv2*f2
_old'+2*f2_old*Qf2f3*f3'+f3*Qfinv3*f3';
```